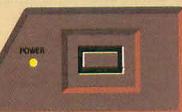
PC-Techknow 9800

テク　ノウ

藤田英時・幸田敏記 共著　システムソフト監修

PCファミリー・テクニカル・ノウハウ集 PC-9800シリーズ編

SystemSoft

PCファミリー テクニカル・ノウハウ集

PC－9800シリーズ編

PC-Techknow9800

共著／藤田英時　幸田敏記
監修／システム ソフト

PC-Techknow9800の発刊にあたって

本書は、ＮＥＣの16ビットパーソナルコンピュータＰＣ－9800シリーズを対象として本体ならびに周辺機器の内部構造から活用の方法までを解説しています。本書の執筆にあたっては、次の点が考慮されています。

●PC－9800シリーズ（PC－9801・PC－9801F・PC－9801E）の３機種に対応するように心掛けています。

●ＰＣ－9800シリーズのバージョンの違いにより本書に記載されているアドレスが異なることがあります。しかし、サンプルプログラムなどは、ほとんどすべてのバージョンに適合するように作られています。これはＲＯＭが異なってもワークエリアは同じであるため、プログラムはワークエリアを参照した形をとっているためです。

●N88－BASIC（86）の内部構造などでのダンプ結果も本書に示されているものと異なっている場合があります。その際は，ワークエリアのアドレスが必ず示されていますので，その値を参照して、ユーザーのマシンに合わせることができます。

●マシン語プログラムのソースリストは内容を理解していただくためにラベルやコメントを付記しています。これらは、デジタルリサーチ社のＡＳＭ－86で作成されています。また、便宜上一部本体モニタのアセンブルリストも用いています。

※ASM－86は米デジタルリサーチ社の登録商標です。
※MS－DOSは米マイクロソフト社の登録商標です。

はじめに

パーソナルコンピュータも８ビットから16ビットの時代になってきました。

16ビットになれば、処理スピード、メモリアクセス能力などが８ビットに比べ飛躍的に向上します。そのためパーソナルコンピュータの利用範囲が拡大し、特にビジネス分野などで効果的にかつ効率良く使われてきております。これからもさらにいろいろな分野で16ビットパーソナルコンピュータが利用されるでしょう。

本書で取り上げるパーソナルコンピュータは、インテルの16ビットCPU、iAPX8086相当を搭載したNECのPC－9800シリーズ(PC－9801・PC－9801F・PC－9801E)です。

これらは、処理スピード、グラフィック機能、メモリ容量など処理能力が、PC－100と共にＰＣシリーズの最上位機種として位置づけられます。

しかし、これまでの８ビットパーソナルコンピュータの仕様とは大幅に異なっていますので、内部構造やメモリアクセスの方法が難解です。PC－9800のユーザーが、パーソナルコンピュータの機能を十二分に引き出し、より有効に使うためには、本体の内部や周辺機器について詳しく知ることがポイントになってきます。そうなればPC－8800シリーズのソフトをPC－9800シリーズで実行させるために移植したり、N_{88}－BASIC(86)やアセンブリ言語でプログラミングする際に効率良く、思いのままの結果が得られることでしょう。

そこで本書では、PC－9800シリーズの本体はもちろん、プリンタ、ディスクユニットに至るまで、内部解析情報や活用のノウハウを実践に役立つようにまとめています。またすぐに使えるプログラムも豊富に紹介しています。

限られたページの中で、充分意を尽すことはできませんが、読者の方々自らの内容補完により、PC－9800 シリーズに対する理解をより深めていただきたいと存じます。また、PC－9800 シリーズを使いこなすための座右の書として利用いただければ幸いです。

本書を執筆するにあたり、日本電気株式会社殿よりハードウェア及びソフトウェアを快く提供していただきましたことを感謝いたします。

なお、本書の執筆にあたっては、藤田英時と幸田敏記が共同で行ない、システムソフト・スタッフが監修を担当しました。

1983年11月

著者

目次

PC-TechKnow 9800の発刊にあたって……2
はじめに……3
第1章　メモリマップ……14
1-1　メモリマップ……14
1-2　アドレスの表わし方……15
1-3　増設RAM(PC-9801-02)……15
1-4　RAMのメモリマップ……17
1-5　ユーザーマシン語の格納法……20
第2章　N_{88}-BASIC(86)の内部構造……21
2-1　プログラムの格納状態……22
2-2　中間言語……24
2-2-1　中間言語コード(0H～7FH)……24
2-2-2　中間言語コード(80H～FFH)……25
2-2-3　中間言語テーブル……25
2-3　ラベルテーブル……33
2-4　変数テーブル……35
2-4-1　単純変数テーブル……36
2-4-2　配列変数テーブル……39
2-5　文字列エリア……44
2-6　BASICプログラム復活法……46
2-7　ステートメント・関数の処理アドレス……51
第3章　テキスト画面……57
3-1　WIDTH……58
3-1-1　WIDTHとDIPスイッチ……58
3-1-2　WIDTH文のパラメータの省略……58
3-2　テキストVRAMのアドレス……59
3-3　画面とアドレスの対応表……59

3-4　アトリビュートエリア……60
3-5　テキスト画面でグラフィックが使える！……61
3-6　画面を縦に２分割……64
3-7　テキスト画面の２ページ目を利用……67
3-8　ひらがなの表示……68
3-9　TABキーとTAB関数……70

第４章　グラフィック画面……73

4-1　G-VRAM……74
　4-1-1　G-VRAMのメモリマップ……74
　4-1-2　高速画面クリア……77
4-2　カラーパレット……77
　4-2-1　マシン語によるカラーパレットの制御……77
　4-2-2　カラーパレットの初期化……80
4-3　ボーダーカラー……80
4-4　グラフィックBIOSとGDC(Graphic Display Controller)……81
　4-4-1　グラフィックBIOSのワークエリア……82
　4-4-2　PSET ドットを打つ……84
　4-4-3　ドットを読み出す……86
　4-4-4　直線・箱型を描く LINE……89
　4-4-5　円弧を描く CIRCLE……93
　4-4-6　グラフィックパターンを描く……98
　4-4-7　高速書き込みモードにする……101
4-5　マシン語によるG-VRAM直接アクセス法……101
4-6　グラフィックLIOとBASICコマンド……104
4-7　3D/パッケージの紹介……121

第５章　キー入力……135

5-1　キー入力バッファ……136
　5-1-1　BIOSキー入力バッファ……136

5-1-2　インタプリタのキー入力バッファ……138
5-2　ファンクションキー……139
5-2-1　ファンクションキーの構造……139
5-2-2　ファンクションキーの初期化……143
5-2-3　ファンクションキーの退避・復活……145
5-3　キー・スキャン方式……146
5-4　キー入力のセンス……150
5-5　キー入力方法・キーセンス比較表……151
第6章　カセットファイル……153
6-1　CMTインターフェイス……154
6-2　CMTとのデータ転送の仕様……154
6-3　データフォーマット……154
6-3-1　プログラムファイル……154
6-3-2　データファイル……155
6-3-3　マシン語ファイル……155
6-4　内部ルーチン……155
6-4-1　データ書き込み……156
6-4-2　データ読み込み……156
6-5　マシン語によるセーブ・ロード……156
6-6　BASICとマシン語を一度にSAVE・LOAD……158
6-7　データの高速セーブ・ロード……161
第7章　ディスクファイル……165
7-1　ディスクファイルの構造……166
7-1-1　ディスク・マップ……166
7-1-2　ディスクアドレスとクラスタとの変換……169
7-1-3　ディレクトリ……169
7-1-4　IDセクタ……171
7-1-5　FAT……171

7-2　DSKF関数……173
7-3　標準ディスク……174
7-3-1　フォーマッティング……174
①物理フォーマット……174
②インターリーブ13とは？……174
③システムフォーマット……175
④サンプル・プログラム……176
⑤読み書きテスト結果……177
7-3-2　ディスクBIOSコマンド……178
①概要……178
②リードID……181
③リードデータ……183
④ライトデータ……185
⑤シーク……186
⑥フォーマット……187
7-4　5インチ・ディスク……189
7-4-1　概要……189
7-4-2　ディスクBIOSコマンド……189
①センス……189
②イニシャライズ……190
③リードデータ……192
④ライトデータ……193
⑤フォーマット……194
⑥片面・両面アクセス……194
7-5　ディスク・ユーティリティ・プログラム……195
7-5-1　ファイルインフォメーション……195
7-5-2　1ファイル転送……197
7-5-3　ファイルネームソート……199
7-5-4　オールマイティディスクダンプ……200
7-5-5　簡易ディスクエディター……202

7-5-6　8インチIDリーダー……203
7-5-7　インテルHEXファイルローダー……205

第8章　プリンタ出力……208

8-1　画面コピー機能……208
8-2　テキスト画面のコピー……211
8-2-1　BASICによるサブルーチン……211
8-2-2　マシン語によるサブルーチン……212
8-3　カラーグラフィックコピー……212
8-3-1　640×200モード……212
8-3-2　640×400モード……215
8-4　アセンブリ言語によるプリンタ出力……217
8-4-1　イニシャライズ……217
8-4-2　1バイト出力……217
8-4-3　複数バイト出力……218
8-5　PRINT／LPRINTあれこれ……219
8-5-1　出力デバイス名の変更……219
8-5-2　切り換えルーチンを作る……220
8-5-3　内部ルーチンを利用する……221
8-5-4　未使用コマンドでの切り換え……222

第9章　漢字……225

9-1　漢字ROMボード……226
9-2　テキスト画面とグラフィック画面に漢字表示……226
9-3　ファンクションキーエリアに漢字表示……227
9-4　漢字フォントパターン読み出し……229
9-5　漢字フォントパターンの拡大表示……235
9-6　漢字・JISコード対応表示……236
9-7　漢字フォントをビットイメージで出力……237
9-8　任意のフォントの作成・出力……242

第10章　USR関数・CALL文とマシン語……247

10-1　マシン語ルーチンの呼び方……248
10-2　マシン語ルーチンの実行と引数の受け渡し方……251
10-2-1　引数がない場合……251
10-2-2　USR関数の引数……252
10-2-3　CALL文の引数……255
10-3　結果の戻し方……261
10-3-1　USR関数の場合……261
10-3-2　CALL文の場合……262
10-4　BASIC+マシン語ルーチン……265
10-4-1　サウンドビープ……265
10-4-2　小文字・大文字変換……266
10-4-3　最大値を求める……267
10-4-4　文字列を逆に表示……268
10-4-5　ROLL200 & ROLL400……269
10-4-6　アドレスサーチ……270

第11章　入出力ファイル……273

11-1　入出力装置とファイル……274
11-2　変数でファイル指定……274
11-3　ファイルバッファ……275
11-4　ファイルバッファ使用例……280
11-5　高速グラフィックス・ローダー……283

第12章　RS-232C……289

12-1　RS-232Cとは……290
12-2　専用ケーブルの作り方……290
12-3　通信モードの指定……291
12-4　プログラムの転送……293
12-4-1　メモリー上にある場合……293

12-4-2　ディスクファイルにある場合……294
12-5　コミュニケーション・プログラム……295

第13章　PC-9801F……297

13-1　システム概要……298
13-2　5インチ倍トラックディスク……298
13-3　漢字ROMと日本語BASIC……298
13-4　拡張グラフィック画面……299
13-5　拡張ステートメント……300
13-6　PC－9801E……305

第14章　ランダムテクニック……307

14-1　行番号0……308
14-2　2バイトの数字を上位・下位の1バイトに分ける……310
14-3　REM文の効率……311
14-4　エラーメッセージをすべて表示するには……312
14-5　マシン語でエラーメッセージを表示……315
14-6　未使用コマンドを使用する……318
14-7　新しいコマンドを作る……326
14-8　8086はリセットがかかったら何処へ？！……330
14-9　INKEY$でカーソル表示……331
14-10　高速リスト……332
14-11　CHR$(13);CHR$(10)とCHR$(13)＋CHR$(10)との違い……333
14-12　OUT PUTとASも変数に使える……333
14-13　キーバッファクリア……334
14-14　リアルタイムで時間表示……335
14-15　モニタモードでファンクションキーを使用する……337

第15章　ユーティリティ……339

15-1　テキストサーチ……340
15-2　リプレイス……347

15-3　バリアブルリスト……359
15-4　バーティカルファイルズ……371

付録……377

付1　機械語プログラム・ソースリスト……379
付2　ROM内ルーチンのINTによる利用……405
(1) INT割込みベクター覧表……406
(2) INT C4Hのソフトウェア　インターフェースの説明……421
（インタプリタ内のルーチンの利用）
付3　ワークエリア一覧表……431
(1) システム共通域……432
(2) BASIC LIOワークエリア……438
(3) シンボルテーブルエリアのワークエリア……452
付4　I／Oポート一覧……453
付5　コマンド・ステートメント・関数処理アドレス一覧表……487
付6　コントロールコード一覧表……489
付7　エラーメッセージ一覧表……493
付8　プリンタ機能一覧表（PC-8821/22 PC-8023）……497
付9　キャラクタコード表……501
付10　USING文フォーマット一覧表……503
付11　Z-80・8086ニーモニック対応表……505

第1章　メモリマップ

1-1　メモリマップ

1-2　アドレスの表し方

1-3　増設RAM(PC-9801-02)

1-4　RAMのメモリマップ

1-5　ユーザーマシン語の格納法

第1章　メモリマップ

1-1　メモリマップ

PC-9801は，標準でRAM 128 Kバイト，グラフィックVRAM 96 Kバイト，テキストVRAM 8 Kバイト，ROM 96 Kバイト，の合計328 Kバイトにも及ぶメモリを持っています。これは，PC-8801に比べると約3倍，PC-8001に比べると約5倍になります。しかも，RAMを512 Kバイト増設することができます。

これらのメモリはCPUが16ビットのμPD 8086のため，PC-8801のようなバンク切り換えをせずとも，主記憶空間に図1-1のように連続して配置することができます。

物理アドレス(16進)

FFFFF
F0000
E8000
E0000
D0000
C0000
B0000
A8000
A0000
90000
80000
70000
60000
50000
40000
30000
20000
10000
00000

N88—BASIC(86)
ROM96Kバイト
システム予備
グラフィック
V-RAM96Kバイト
テキストVRAM12Kバイト，不揮発性メモリ8バイト
増設RAMボード
128Kバイト×2
増設RAMボード
128Kバイト×2
標準実装
128Kバイト
(ワークエリア
テキストRAM)
割り込みベクトル
部分が標準実装されています。

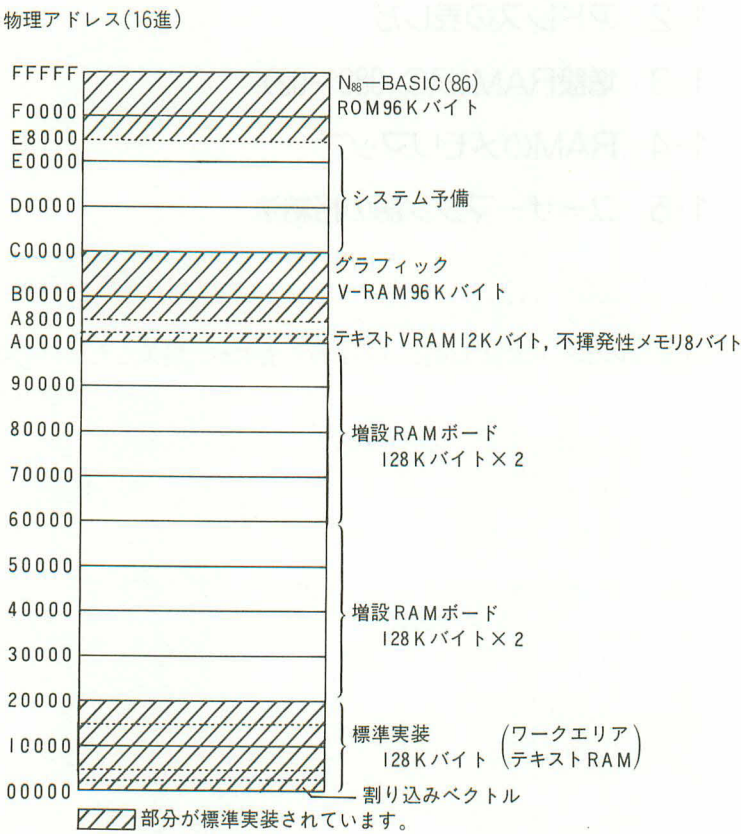



図1-1 全アドレス空間の概要

1-2　アドレスの表し方

アドレスは実質的に，20 ビット＝4 ビット×5，即ち 5 ケタの 16 進数字で表わします。このアドレスのことを物理アドレスといい 1 M バイトまで表現できます。

8086 は，セグメントという考え方を導入して，1 つのセグメント内では，一度に 64 K バイトまでしかアクセスできません。このアドレスのことをオフセットと呼びます。即ち，図 1-2 のように，セグメントアドレスを 4 ビット左へずらして，オフセットアドレスを加えたものが物理アドレスということになります。普通は，セグメントアドレスを固定して，オフセットアドレスで指定することになります。

□の中の数字は例として示したものです。

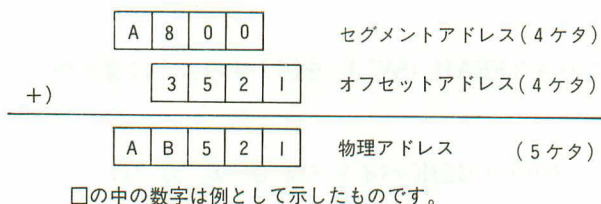



図 1-2 アドレス表現

1-3　増設RAM（PC-9801-02）

RAM を増設した際 1 つのボードには 256 K バイトのりますが，購入した時点では，128 K バイトしかのっていませんので注意して下さい。256 K バイトにするには，PC-9805 増設用 RAM 128 K バイトをボードの空ソケットに差さなければなりません。しかし，RAM ボードを差し込んだだけではメモリは増えません。本体背面のディップスイッチ SW 2 の 5 を ON（下向き）にして，主メモリ内の物理アドレス A 3 FE 0 H～A 3 FFFH の 32 バイトに 4 バイトおきに配置された不揮発性メモリを書きかえなければなりません。それぞれのメモリスイッチには図 1-3 のようにSW 1～SW 7 の名前がついています。

物理アドレス	名　前
A 3 F E 2	S W 1
A 3 F E 6	S W 2
A 3 F E A	S W 3
A 3 F E E	S W 4
A 3 F F 2	S W 5
A 3 F F 6	S W 6
A 3 F F A	S W 7

図 1-3 メモリ・スイッチの名前

メモリ増設のときは，このA 3 FEAH（SW 3）を図 1-4 のように書きかえなければなりません。

0………128K バイト（標 準 実 装 時）
1………256K バイト（128K バイト増設）
2………384K バイト（256K バイト増設）
3………512K バイト（384K バイト増設）
4………640K バイト（512K バイト増設）

図 1-4 メモリ増設のときの SW 3 の設定

ただしこれは，A 3 FEAH の下位 3 ビット(b_0, b_1, b_2)であって，上位 5 ビットは次の表 1-1 の意味をもちますので，ユーザーが自分のマシンに合わせて設定します。

	0	1
b_3………8087	無	有
b_4………ODA 系プリンタ	JIS 8 ビット	7 ビット
b_5………プリンタの型	セントロニクス	ODA 系
b_6………テキスト画面のカラー	白	緑
b_7………DEL コード受信時動作	BS	NUL

表 1-1 SW 3 のユーザー設定

普通の使い方（8087 なし，プリンタはセントロニクス，文字の色は白）なら，図 1-4 の値を入れておけば OK です。

設定の方法は，MON モードにして，

```
h ] SSW3 ←┐      (128Kバイト増設時)
00 - 01 ←┘(この部分を入力する。)
```

というように書きかえます。

h] SSW

と入力すると，全メモリスイッチの内容が表示されます。リセットをすると，この場合128Kバイト増設した状態で使えます。もし，このセッティングをしないと，せっかくRAMを増設してもBASICのフリーエリアは増えませんので注意して下さい。

1-4 RAMのメモリマップ

図1-1にアドレス空間全体のメモリマップを示しました。今度は，さらに細かくみていくことにしましょう。標準実装RAMは，物理アドレス00000H～1FFFFHに配置されています。N_{88}-BASIC(86)使用時のメモリ・マップを図1-5に示します。

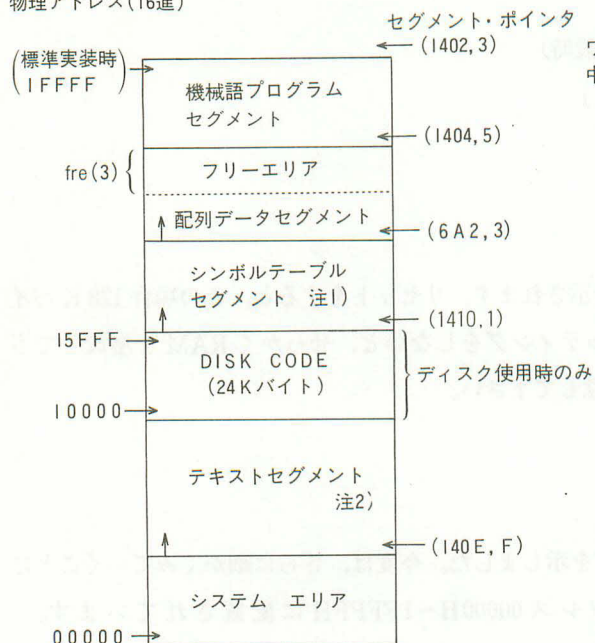
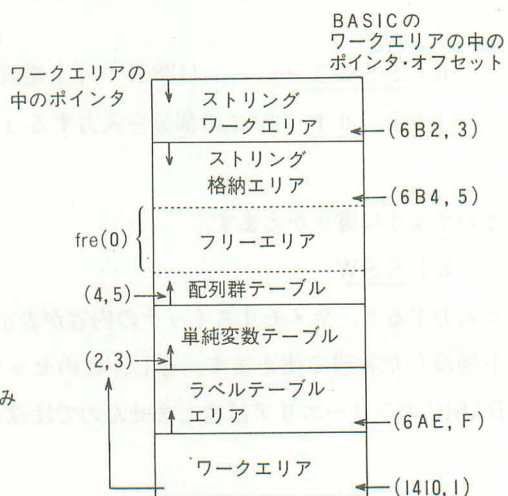
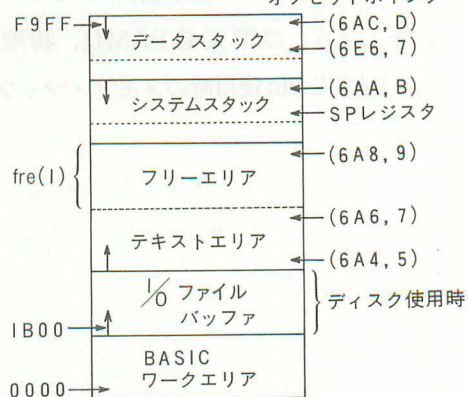



図 1-5 N₈₈-BASIC(86) 使用時のメモリマップ

表1-2は，図1-5の各セグメントポインタの値です。

	ポインタアドレス	内　　容	
		標　準	128KB増
テキストのセグメントの値	140E,0F	0060	0060
シンボルテーブルのセグメント値	1410,11	DISK　1600 ROM　1000	1600
配列データセグメント	6A2,3	DISK　1B00 ROM　1800	DISK　25FF ROM　1FFF
マシン語のセグメント値 (CLEAR文の第2引数で設定した値)	1404,05	2000	4000
RAM終端物理アドレス＋1のセグメント値	1402,03	2000	4000

表1-2 セグメント・ポインタの内容（16進）

これらのポインタはセグメント60Hからのオフセットです。表1-2の内容は，MONモードで次のようにして調べることができます。

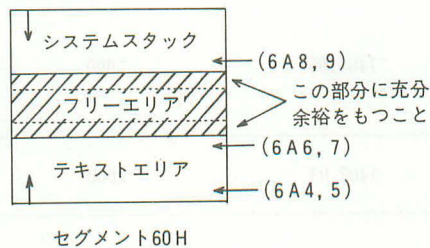
```
MON
h]C60
h]D1402
1402 00 20 00 20 02 02 00 1B 38 06 00 E8 60 00 00 16             8  ♠
h]D6A2
06A2 00 1B D8 27 C3 28 FE F3 FE F7 FE F9 00 01 00 01          リ´テ( 月 秒
h]
```

マシン語のセグメント値は電源ON(リセット)の時点ではRAM終端と同じですが，CLEARの第2引数の値で指定することができます。

図1-5の中でテキストセグメントの中にI/Oファイルバッファという部分がありますが，この中にFAT等を格納するエリアがあります。詳細については『第7章 ディスクファイル』で説明します。

データスタックは，GOSUB，FOR，WHILEのネスティングの記憶に使われます。それぞれ1回につき8バイト，22バイト，10バイトずつ消費されます。データスタックの大きさは初期設定では512バイトですので，ネスティングを深くするときは，データスタックの大きさをふやして下さい。この大きさは，CLEAR文の第3引数で指定することができます。

1-5　ユーザーマシン語の格納法

ユーザーのマシン語の格納法には2種類あります。1つはマニュアルに記されているように，CLEAR宣言の第2引数でマシン語エリアの先頭セグメントを指定する方法です。もう1つは，テキストの空エリアを利用する方法です。テキストのエンドポインタは，6A6H，6A7Hに格納されています。セグメントは，セグメントポインタ140EH，140FHの中に格納されているセグメント値(通常60H)を使用します。上限のオフセットはポインタ6A8H，6A9Hに格納されていますので，充分な余裕を取って使用すればよいでしょう。ただし，PAINTを同時に使用することはできません。それはPAINTルーチンが作業領域としてこの部分を使用するからです。

第2章　N88-BASIC(86)の内部構造

2-1　プログラムの格納状態

2-2　中間言語

2-2-1　中間言語コード(0H～7FH)
2-2-2　中間言語コード(80H～FFH)
2-2-3　中間言語テーブルの形式

2-3　ラベルテーブル

2-4　変数テーブル

2-4-1　単純変数テーブル
2-4-2　配列変数テーブル

2-5　文字列エリア

2-6　BASICプログラム復活法

2-7　ステートメント・関数の処理アドレス

第2章　N88-BASIC(86)の内部構造

2-1　プログラムの格納状態

BASICプログラムは，テキストエリアの先頭から，図2-1のような形式で格納されていきます。

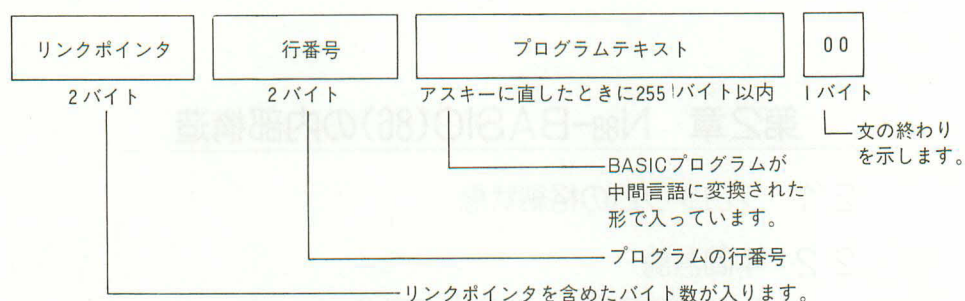



図2-1 テキストの格納形式

リンクポインタは，PC-8001，PC-8801と違って，アドレスではなく，その行の占めるバイト数となっています。プログラムの終わりでは，リンクポインタの値が0となります。

次に，プログラムがどのように格納されているか見てみましょう。例として，次のプログラムを使います。

```
10 REM TEST FOR TEXT
20 A=0
30 FOR I=1 TO 10:A=A+1:NEXT
40 PRINT A
50 END
```

テキストのTOP，ENDはセグメント60Hのオフセットポインタ6A4H，6A5Hと6A6H，6A7Hに格納されていますので，

```
MON
hJC60
hJD6A4,6A7
06A4 48 23 96 23
```

テキスト END（セグメント60Hからのオフセット）　このオフセットは、DISK使用によって変化します。

テキスト TOP

```
hJD2348,2367
2348 18 00 0A 00 01 00 52 45 4D 20 54 45 53 54 20 46
2358 4F 52 20 54 45 58 54 00 0A 00 14 00 01 41 00 F1
2368 10 00 1B 00 1E 00 01 9E 01 49 00 F1 11 01 DA 01
2378 0F 0A 3A 41 00 F1 41 00 F3 11 3A B7 00 0A 00 28
2388 00 01 C0 01 41 00 00 07 00 32 00 01 9A 00 00 00
```

とダンプして直接見ることができます。

各データは，図 2-2 のような意味を持ちます。

```
2348 18 ┐リンクポインタ →2360 0A ┐リンクポインタ →236A 1B ┐リンクポインタ →2385 0A ┐リンクポインタ
2349 00 ┘2348+18          2361 00 ┘2360+0A          236B 00 ┘236A+1B          2386 00 ┘2385+0A
234A 0A ┐行番号           2362 14 ┐行番号           236C 1E ┐行番号           2387 28 ┐行番号
234B 00 ┘ (=10)           2363 00 ┘(=20)            236D 00 ┘(=30)            2388 00 ┘(=40)
234C 01 ]スペース1コ      2364 01 ]スペース1コ      236E 01 ]スペース1コ      2389 01 ]スペース1コ
234D 00 ┐                 2365 41 ┐A…変数名         236F 9E ]FOR              238A C0 ]PRINT
234E 52 │R                2366 00 ┘                 2370 01 ]スペース1コ      238B 01 ]スペース1コ
234F 45 │E                2367 F1 ]=                2371 49 ┐I…変数名         238C 41 ┐A
2350 4D ┘M                2368 10 ]0                2372 00 ┘                 238D 00 ┘
2351 20 ]スペース         2369 00 ]行の終わり       2373 F1 ]=                238E 00 ]行の終わり
2352 54 ┐T                                          2374 11 ]1
2353 45 │E                                          2375 01 ]スペース1コ
2354 53 │S                                          2376 DA ]TO
2355 54 ┘T                                          2377 01 ]スペース1コ
2356 20 ]スペース                                   2378 0F ┐10
2357 46 ┐F                                          2379 0A ┘
2358 4F │O                                          237A 3A ]マルチステートメント
2359 52 ┘R                                          237B 41 ┐A
235A 20 ]スペース                                   237C 00 ┘
235B 54 ┐T                                          237D F1 ]=
235C 45 │E                                          237E 41 ┐A
235D 58 │X                                          237F 00 ┘
235E 54 ┘T                                          2380 F3 ]+
235F 00 ]行の終わり                                 2381 11 ]1
                                                    2382 3A ]:
                                                    2383 B7 ]NEXT
                                                    2384 00 ]行の終わり

→238F 07 ┐リンクポインタ      →2396 00 ┐プログラムの終わり
 2390 00 ┘238F+7               2397 00 ┘
 2391 32 ┐行番号
 2392 00 ┘(=50)
 2393 01 ]スペース1コ
 2394 9A ]END
 2395 00 ]行の終わり
```

図 2-2 プログラムの格納状態サンプル

2-2 中間言語

前の節で，BASICプログラムのテキストが中間言語を使って，短縮された形で格納されていることがわかりました。

それでは，N88-BASIC(86)で，どのような中間言語が使われているのかを見てみましょう。

2-2-1 中間言語コード（0H〜7FH）

中間言語コードの0Hから7FHは，数値や行番号，変数名に使われます。今までと大きく違っているところは，1〜9がスペースの数となっているところで，BASICテキストの段づけなどをしたときのメモリの節約になっています（図2-2のサンプルには，01スペース1コというところがありますが，ここを05とすると，スペースが5コになります）。

これらをまとめたものを図2-3に示します。

中間言語		意味	備考
0	REM エンドマーク	そこからあとはREMと同じあつかい（文の途中），または，行の終わり	
1〜9	スペース	スペース1〜9コ	
0A	LF	Line Feed	CTRL + J で入力する
0B	&O	以下の2バイトは8進数	0B 9C 02 = &O1234
0C	&H	以下の2バイトは16進数	0C 34 12 = &H1234
0D	アドレス	以下の2バイトは飛び先オフセットアドレス	GOTO, GOSUB, THEN, ELSE, RESTORE等の後に続きます。
0E	行番号	以下の2バイトは飛び先行番号	
0F	整数	以下の1バイトは，10〜255の整数	0F 50 = 80(10)
10〜19	整数	1桁の整数 （10→0，11→1，……，19→9）	
1A		使われていない。Syntax error になる。	
1B		漢字のシフトイン，シフトアウト	
1C	整数	以下の2バイトは，整数	1C D2 04 = 1234
1D	単精度	以下の4バイトは，単精度定数	1D EB C0 1D 81 = 1.2345
1F	倍精度	以下の8バイトは，倍精度定数	
20 〜 7F	文字	キャラクタコードに対応する文字 （変数名やラベル名など）	DATA文やREM文，クォテーションの中以外では，アルファベットの小文字は大文字に変換されるため使われません。上記文の中のコードはインタプリタは中間コードとしては実行しません。

図2-3 中間言語コード（0H〜7FH）

2-2-2 中間言語コード（80 H～FFH）

中間言語コードの80 HからFFHは1バイトまたは2バイトで，N_{88}-BASIC(86)のキーワードを示します（表2-1〔P 30〕～表2-2〔P 31〕)。

N-BASICやN_{88}-BASICと比べると，中間言語コードが異なります。したがって，バイナリでSAVEしたBASICファイルをそのままLOADしても正常に動作させることはできません。N_{88}-BASICのものであれば，アスキーSAVE

SAVE〝ファイル名″，A

を行ったものであれば，USRやPEEK，POKEなどを使っていなければ，動作します。

表2-1，表2-2で(　)でくくってある数字は，フラグとよばれるもので図2-4に示すような意味をもっています。これはインタプリタ内での処理を容易にするために新しく設けられたもののようです。

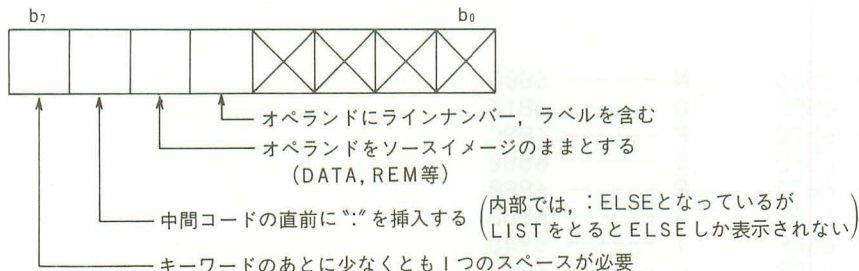



図2-4 キーワードテーブル中のフラグの意味

2-2-3 中間言語テーブル

中間言語とキーワードの対応表は，セグメントE 800 H，オフセット0000 H～B 1 FFHのどこかに位置していますが，ROMのバージョン等によって相当異なります。ただし，テーブルのオフセットを格納しているアドレスをBASICのワークエリア（セグメント60 H）の140 AH～140 DHに，オフセットセグメントの順に格納してありますので，ROMのバージョンが異なってもキーワード・テーブルの位置を知ることができます。

このテーブルは，入力したプログラムを中間言語に変換してテキスト領域に格納するときや，逆にリストをとったりするときに使われるものです。キーワードテーブルのアドレス値は，ROMのバージョンによって異なりますが，一例を次に示します。

```
h]C60
h]D140A,140D
140A FE 64 00 E8
h]
```

セグメントE800H，オフセット64FEHにキーワードテーブルのポインタがあることを示しています。

```
h]CE800
h]D64FE,6535
64FE 36 65 5F 65 73 65 FD 65 50 66 88 66 AF 66 C8 66   6e_ese ePfl fｯfﾈf
650E D4 66 19 67 1F 67 6A 67 CB 67 00 68 16 68 3C 68   ﾔf g gjgﾋg h h<h
651E 80 68 80 68 DB 68 45 69 73 69 7F 69 92 69 C2 69   _h_hﾛhEisi iｲiﾂi
652E C7 69 C7 69 C7 69 EF 69                           ﾇiﾇiﾇi\i
```

これは，頭文字のインデックスポインタ群です。

```
A ------ 6536      N ------ 6800
B ------ 655F      O ------ 6816
C ------ 6573      P ------ 683C
D ------ 65FD      Q ------ 6880
E ------ 6650      R ------ 6880
F ------ 6688      S ------ 68DB
G ------ 66AF      T ------ 6945
H ------ 66C8      U ------ 6973
I ------ 66D4      V ------ 697F
J ------ 6719      W ------ 6992
K ------ 671F      X ------ 69C2
L ------ 676A      Y ------ 69C7
M ------ 67CB      Z ------ 69C7
```

ということであり，

```
MON
h]D6536,655F
6536 03 55 54 4F 80 90 02 4E 44 F8 80 02 42 53 10 80   UTO_ｰ ND _ BS _
6546 02 54 4E 11 80 02 53 43 12 80 04 54 54 52 24 13   TN _ SC _ TTR$
6556 00 05 4B 43 4E 56 24 4C 00 04                      KCNV$L
```

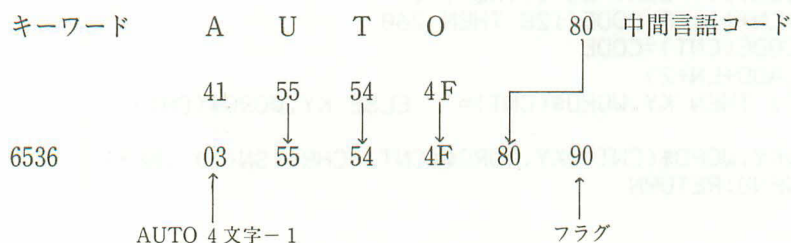
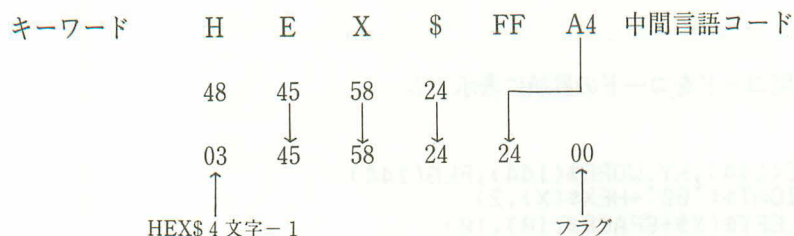
例えば，6536H～655FHに格納されているキーワードは頭文字がAということを示すものです。

AUTO，AND，ABS，ATN，ASC，ATTR$，AKCNV$

が右側のアスキーダンプにみえています。

2-2-4 中間言語テーブルの形成

中間言語は，キーワードの１文字目によって，アルファベット順に分類されています。各データは，キーワードの文字数−1，キーワード，中間言語，フラグから成り立っています。キーワードのデータは１文字目が省略され(１文字目でグループ分けしてあるので不要)，そのかわりキーワードの先頭にキーワードの文字数−1を示す１バイトの数字があります。また，中間コードについては，１バイトのものはそのままの形で，２バイトのもの（FF+XX）は，最上位ビットを０にして１バイトで表わせるようにしてあります。

〈中間言語コードが１バイト〉

〈中間言語コードが２バイト〉

次のプログラムで，これらのデータ(キーワードと中間言語，及び，フラグ)を出力してみましょう。リスト2-1が表2-1，リスト2-2が表2-2，リスト2-3が表2-3を出力するものです。

リストの先頭が０行になっていますが，これは，入力しなくてよいです。０行の作り方は第14章ランダムテクニックでお教えします……。

リスト2-1　1バイトの中間コードをコードの昇順に表示する

```
0 'SAVE"KEY1"
10 DIM SN(10),CODE(144),KY.WORD$(144),FLG(144)
20 DEF FNHX$(X)=RIGHT$("00"+HEX$(X),2)
30 DEF FNWD$(X$)=LEFT$(X$+SPACE$(10),10)
40 DEF FNKY$(X)=FNHX$(I+&H80+X)+"("+FNHX$(FLG(I+X))+") : "
```

```
   +FNWD$(KY.WORD$(I+X))+"    "
50 DEF SEG=&H60
60 AD=PEEK(&H140A)+256*PEEK(&H140B):'GET KEY WORD TABLE POINTER
70 BSE=PEEK(&H140C)+256*PEEK(&H140D):' GET SEGMENT
80 CHR=ASC("A"):DEF SEG=BSE
90 WHILE CHR<92
100 ADD=PEEK(AD)+256*PEEK(AD+1)     :' CURRENT ALPHABET ADDRESS
110 NADD=PEEK(AD+2)+256*PEEK(AD+3):' NEXT ALPHABET ADDRESS
120 GOSUB *LST                      :' GET KEY WORD LIST
130 AD=AD+2:CHR=CHR+1               :' NEXT ALPHABET
140 WEND
150 GOTO *PRT.OUT                   :' PRINT OUT KEY WORDS
160 END
170 *LST
180 WHILE ADD<NADD
190 LN=PEEK(ADD)
200 FOR I=1 TO LN:SN(I)=PEEK(ADD+I):NEXT I
210 CODE=PEEK(ADD+LN+1):IF CODE<128 THEN 260
220 CNT=CODE-128:CODE(CNT)=CODE
230 FLG(CNT)=PEEK(ADD+LN+2)
240 IF CHR=ASC("[") THEN KY.WORD$(CNT)="" ELSE KY.WORD$(CNT)
    =CHR$(CHR)
250 FOR I=1 TO LN:KY.WORD$(CNT)=KY.WORD$(CNT)+CHR$(SN(I)):NEXT
260 ADD=ADD+LN+3:WEND:RETURN
270 *PRT.OUT
280 FOR I=0 TO 47
290 PRINT FNKY$(0);
300 IF I+&HB0<256 THEN PRINT FNKY$(&H30);
310 IF I+&HE0<256 THEN PRINT FNKY$(&H60) ELSE PRINT
320 NEXT
```

リスト2-2 2バイトの中間コードをコードの昇順に表示する

```
0 'SAVE"KEY2"
10 DIM SN(10),CODE(144),KY.WORD$(144),FLG(144)
20 DEF FNHX$(X)=RIGHT$("00"+HEX$(X),2)
30 DEF FNWD$(X$)=LEFT$(X$+SPACE$(10),10)
40 DEF FNKY$(X)="FF  "+FNHX$(&H80+I+X)+"("+FNHX$(FLG(I+X))+") :
   "+FNWD$(KY.WORD$(I+X))+"    "
50 DEF SEG=&H60
60 AD=PEEK(&H140A)+256*PEEK(&H140B):'GET KEY WORD TABLE POINTER
70 BSE=PEEK(&H140C)+256*PEEK(&H140D):' GET SEGMENT
80 CHR=ASC("A"):DEF SEG=BSE         : ' SET THE SEGMENT
90 WHILE CHR<92
100 ADD=PEEK(AD)+256*PEEK(AD+1)     :' CURRENT ALPHABET ADDRESS
110 NADD=PEEK(AD+2)+256*PEEK(AD+3):' NEXT ALPHABET ADDRESS
120 GOSUB *LST                      :' GET KEY WORD LIST
130 AD=AD+2:CHR=CHR+1               :' NEXT ALPHABET
140 WEND
150 GOTO *PRT.OUT                   :' PRINT OUT KEY WORDS
160 END
170 *LST
180 WHILE ADD<NADD
190 LN=PEEK(ADD)
200 FOR I=1 TO LN:SN(I)=PEEK(ADD+I):NEXT I
210 CODE=PEEK(ADD+LN+1):IF CODE>127 THEN 260
220 CNT=CODE:CODE(CNT)=CODE
230 FLG(CNT)=PEEK(ADD+LN+2)
```

```
240 IF CHR=ASC("[") THEN KY.WORD$(CNT)="" ELSE KY.WORD$(CNT)
    =CHR$(CHR)
250 FOR I=1 TO LN:KY.WORD$(CNT)=KY.WORD$(CNT)+CHR$(SN(I)):NEXT
260 ADD=ADD+LN+3:WEND:RETURN
270 *PRT.OUT
280 FOR I=0 TO 47
290 PRINT FNKY$(0);
300 IF I+&HB0<256 THEN PRINT FNKY$(&H30);
310 IF I+&HE0<256 THEN PRINT FNKY$(&H60) ELSE PRINT
320 NEXT
```

リスト2-3 キーワードの頭文字の昇順に表示する

```
0 'SAVE"KEY3"
10  DIM SN(10),KY$(255),CODE(255),FLG(255)
20   DEF FNHX$(X)=RIGHT$("00"+HEX$(X),2)
30   DEF FNWD$(X$)=LEFT$(X$+SPACE$(10),10)
40   DEF SEG=&H60
50  AD=PEEK(&H140A)+256*PEEK(&H140B):'GET KEY WORD TABLE POINTER
60  BSE=PEEK(&H140C)+256*PEEK(&H140D):' GET SEGMENT
70  CHR=ASC("A"):DEF SEG=BSE:COUNT=-1: ' SET THE SEGMENT
80  WHILE CHR<92
90   ADD=PEEK(AD)+256*PEEK(AD+1)    :' NOW  ALPHABET ADDRESS
100   NADD=PEEK(AD+2)+256*PEEK(AD+3):' NEXT ALPHABET ADDRESS
110    GOSUB *LST                    :' GET KEY WORD LIST
120  AD=AD+2:CHR=CHR+1               :' NEXT ALPHABET
130 WEND
140  GOTO *PRT.OUT
150 END
160 *LST
170   WHILE ADD<NADD
180    LN=PEEK(ADD):COUNT=COUNT+1
190     FOR I=1 TO LN:SN(I)=PEEK(ADD+I):NEXT I
200      CODE(COUNT)=PEEK(ADD+LN+1)
210      FLG(COUNT)=PEEK(ADD+LN+2)
220      IF CHR=ASC("[") THEN KY$(COUNT)="" ELSE KY$(COUNT)=CHR$(CHR)
230     FOR I=1 TO LN:KY$(COUNT)=KY$(COUNT)+CHR$(SN(I)):NEXT
240    ADD=ADD+LN+3
250   WEND
260  RETURN
270 *PRT.CODE
280  IF CODE<&H80 THEN PRINT "FF ";
290   PRINT FNHX$(CODE OR &H80)" ";
300  RETURN
310 *PRT.OUT
320  FOR J=0 TO 66
330   I=J:TB=18:GOSUB *PRT.SUB
340     I=J+67:TB=42:GOSUB *PRT.SUB
350    I=J+67*2:IF I>255 THEN PRINT:GOTO 370
360   TB=67:GOSUB *PRT.SUB:PRINT
370  NEXT
380 END
390 *PRT.SUB
400  IF ASC(FNWD$(KY$(I)))=32 THEN 440
410  CODE=CODE(I):PRINT FNWD$(KY$(I))" ";
420   GOSUB *PRT.CODE
430  PRINT TAB(TB)"("FNHX$(FLG(I))")    ";
440 RETURN
```

80(90) : AUTO	B0(80) : LINE	E0(80) : WHILE
81(80) : BSAVE	B1(80) : LOAD	E1(80) : WEND
82(80) : BLOAD	B2(80) : LSET	E2(80) : WRITE
83(80) : BEEP	B3(80) : LFILES	E3(90) : LIST
84(80) : CONSOLE	B4(80) : MOTOR	E4(80) : SEG
85(80) : COPY	B5(80) : MERGE	E5(80) : SET
86(80) : CLOSE	B6(80) : MON	E6(80) : KINPUT
87(80) : CONT	B7(80) : NEXT	E7(80) : SRQ
88(80) : CLEAR	B8(80) : NAME	E8(80) : CMD
89(80) : CALL	B9(80) : NEW	E9(80) : IRESET
8A(80) : COMMON	BA(80) : NOT	EA(80) : ISET
8B(80) : CHAIN	BB(80) : OPEN	EB(80) : POLL
8C(80) : COM	BC(80) : OUT	EC(80) : RBYTE
8D(80) : CIRCLE	BD(80) : ON	ED(80) : WBYTE
8E(80) : COLOR	BE(80) : OPTION	EE(80) : KPLOAD*
8F(80) : CLS	BF(80) : OFF	EF(00) :
90(90) : DELETE	C0(00) : ?	F0(00) : >
91(A0) : DATA	C1(80) : PUT	F1(00) : =
92(80) : DIM	C2(80) : POKE	F2(00) : <
93(80) : DEFSTR	C3(80) : PSET	F3(00) : +
94(80) : DEFINT	C4(80) : PRESET	F4(00) : -
95(80) : DEFSNG	C5(80) : PAINT	F5(00) : *
96(80) : DEFDBL	C6(90) : RETURN	F6(00) : /
97(00) : DSKO$	C7(80) : READ	F7(00) : ^
98(80) : DEF	C8(90) : RUN	F8(80) : AND
99(D0) : ELSE	C9(90) : RESTORE	F9(80) : OR
9A(80) : END	CA(00) :	FA(80) : XOR
9B(80) : ERASE	CB(90) : RESUME	FB(80) : EQV
9C(90) : EDIT	CC(80) : RSET	FC(80) : IMP
9D(80) : ERROR	CD(90) : RENUM	FD(80) : MOD
9E(80) : FOR	CE(80) : RANDOMIZE	FE(00) : ¥
9F(80) : FIELD	CF(80) : ROLL	FF(80) : REM
A0(80) : FILES	D0(80) : SCREEN	
A1(00) : FN	D1(80) : STOP	
A2(80) : DRAW*	D2(80) : SWAP	
A3(90) : GO TO	D3(80) : SAVE	
A4(90) : GOSUB	D4(80) : SPC	
A5(80) : GET	D5(80) : STEP	
A6(80) : HELP	D6(90) : THEN	
A7(80) : INPUT	D7(80) : TRON	
A8(80) : IF	D8(80) : TROFF	
A9(80) : KEY	D9(80) : TAB	
AA(80) : KILL	DA(80) : TO	
AB(80) : KANJI	DB(80) : TERM	
AC(80) : LOCATE	DC(80) : USING	
AD(00) : L?	DD(00) : USR	
AE(90) : LLIST	DE(80) : WIDTH	
AF(80) : LET	DF(80) : WAIT	

注）＊印はPC-9801F･Eで追加されたもの

注）REMの中間コードはFFになっていますが，実際にはこれが使われず，

00　52　45　40

N_L　R　E　M

の形で格納されます。

文の途中で，中間コード0があらわれるとインタプリタはそこからあとはREM文として処理します。

表2-1　1バイトで表わされる中間コード（　）の中はフラグ

```
FF 80(00) : DATE$      FF B0(00) : MKS$       FF E0(00) :
FF 81(00) : MID$       FF B1(00) : MKD$       FF E1(00) :
FF 82(80) : POINT      FF B2(80) : MAP        FF E2(00) :
FF 83(80) : PEN        FF B3(00) : OCT$       FF E3(00) :
FF 84(00) : TIME$      FF B4(80) : POS        FF E4(00) :
FF 85(80) : VIEW       FF B5(80) : PEEK       FF E5(00) :
FF 86(80) : WINDOW     FF B6(00) : RIGHT$     FF E6(00) :
FF 87(00) :            FF B7(80) : RND        FF E7(00) :
FF 88(00) :            FF B8(80) : SEARCH     FF E8(00) :
FF 89(00) :            FF B9(80) : SGN        FF E9(00) :
FF 8A(00) :            FF BA(80) : SQR        FF EA(00) :
FF 8B(00) :            FF BB(80) : SIN        FF EB(00) :
FF 8C(00) :            FF BC(00) : STR$       FF EC(00) :
FF 8D(00) :            FF BD(00) : STRING$    FF ED(00) :
FF 8E(00) :            FF BE(00) : SPACE$     FF EE(00) :
FF 8F(00) :            FF BF(80) : TAN        FF EF(00) :
FF 90(80) : ABS        FF C0(80) : VAL        FF F0(00) :
FF 91(80) : ATN        FF C1(00) : DSKI$      FF F1(00) :
FF 92(80) : ASC        FF C2(80) : FRE        FF F2(00) :
FF 93(00) : ATTR$      FF C3(80) : VARPTR     FF F3(00) :
FF 94(80) : CSRLIN     FF C4(00) : INPUT$     FF F4(00) :
FF 95(80) : CINT       FF C5(00) : JIS$       FF F5(00) :
FF 96(80) : CSNG       FF C6(00) : KNJ$       FF F6(00) :
FF 97(80) : CDBL       FF C7(80) : KTYPE      FF F7(00) :
FF 98(80) : CVI        FF C8(80) : KLEN       FF F8(00) :
FF 99(80) : CVS        FF C9(00) : KMID$      FF F9(00) :
FF 9A(80) : CVD        FF CA(00) : KEXT$      FF FA(00) :
FF 9B(80) : COS        FF CB(80) : KINSTR     FF FB(00) :
FF 9C(00) : CHR$       FF CC(00) : AKCNV$     FF FC(00) :
FF 9D(80) : DSKF       FF CD(00) : KACNV$     FF FD(00) :
FF 9E(90) : ERL        FF CE(80) : IEEE       FF FE(00) :
FF 9F(80) : ERR        FF CF(80) : STATUS     FF FF(00) :
FF A0(80) : EXP        FF D0(00) :
FF A1(80) : EOF        FF D1(00) :
FF A2(80) : FIX        FF D2(00) :
FF A3(80) : FPOS       FF D3(00) :
FF A4(00) : HEX$       FF D4(00) :
FF A5(80) : INSTR      FF D5(00) :
FF A6(80) : INT        FF D6(00) :
FF A7(80) : INP        FF D7(00) :
FF A8(00) : INKEY$     FF D8(00) :
FF A9(80) : LPOS       FF D9(00) :
FF AA(80) : LOG        FF DA(00) :
FF AB(80) : LOC        FF DB(00) :
FF AC(80) : LEN        FF DC(00) :
FF AD(00) : LEFT$      FF DD(00) :
FF AE(80) : LOF        FF DE(00) :
FF AF(00) : MKI$       FF DF(00) :
```

表2-2 2バイトで表わされる中間コード（ ）の中はフラグ

AUTO	80	(90)	HELP	A6	(80)	PRESET	C4	(80)
AND	F8	(80)	INPUT$	FF C4	(00)	POINT	FF 82	(80)
ABS	FF 90	(80)	INPUT	A7	(80)	PAINT	C5	(80)
ATN	FF 91	(80)	IF	A8	(80)	PEN	FF 83	(80)
ASC	FF 92	(80)	INSTR	FF A5	(80)	POLL	EB	(80)
ATTR$	FF 93	(00)	INT	FF A6	(80)	RETURN	C6	(90)
AKCNV$	FF CC	(00)	INP	FF A7	(80)	READ	C7	(80)
BSAVE	81	(80)	IMP	FC	(80)	RUN	C8	(90)
BLOAD	82	(80)	INKEY$	FF A8	(00)	RESTORE	C9	(90)
BEEP	83	(80)	IEEE	FF CE	(80)	REM	FF	(80)
CONSOLE	84	(80)	IRESET	E9	(80)	RESUME	CB	(90)
COPY	85	(80)	ISET	EA	(80)	RSET	CC	(80)
CLOSE	86	(80)	JIS$	FF C5	(00)	RIGHT$	FF B6	(00)
CONT	87	(80)	KEY	A9	(80)	RND	FF B7	(80)
CLEAR	88	(80)	KILL	AA	(80)	RENUM	CD	(90)
CSRLIN	FF 94	(80)	KANJI	AB	(80)	RANDOMIZE	CE	(80)
CINT	FF 95	(80)	KINPUT	E6	(80)	ROLL	CF	(80)
CSNG	FF 96	(80)	KNJ$	FF C6	(00)	RBYTE	EC	(80)
CDBL	FF 97	(80)	KTYPE	FF C7	(80)	SCREEN	D0	(80)
CVI	FF 98	(80)	KLEN	FF C8	(80)	SEARCH	FF B8	(80)
CVS	FF 99	(80)	KMID$	FF C9	(00)	STOP	D1	(80)
CVD	FF 9A	(80)	KEXT$	FF CA	(00)	SWAP	D2	(80)
COS	FF 9B	(80)	KINSTR	FF CB	(80)	SAVE	D3	(80)
CHR$	FF 9C	(00)	KACNV$	FF CD	(00)	SPC	D4	(80)
CALL	89	(80)	KPLOAD	EE	(80)	STEP	D5	(80)
COMMON	8A	(80)	LOCATE	AC	(80)	SGN	FF B9	(80)
CHAIN	8B	(80)	LPRINT	AD	(80)	SQR	FF BA	(80)
COM	8C	(80)	L?	AD	(00)	SIN	FF BB	(80)
CIRCLE	8D	(80)	LIST	E3	(90)	STR$	FF BC	(00)
COLOR	8E	(80)	LLIST	AE	(90)	STRING$	FF BD	(00)
CLS	8F	(80)	LPOS	FF A9	(80)	SPACE$	FF BE	(00)
CMD	E8	(80)	LET	AF	(80)	SEG	E4	(80)
DELETE	90	(90)	LINE	B0	(80)	SET	E5	(80)
DATA	91	(A0)	LOAD	B1	(80)	STATUS	FF CF	(80)
DIM	92	(80)	LSET	B2	(80)	SRQ	E7	(80)
DEFSTR	93	(80)	LFILES	B3	(80)	THEN	D6	(90)
DEFINT	94	(80)	LOG	FF AA	(80)	TRON	D7	(80)
DEFSNG	95	(80)	LOC	FF AB	(80)	TROFF	D8	(80)
DEFDBL	96	(80)	LEN	FF AC	(80)	TAB	D9	(80)
DSKO$	97	(00)	LEFT$	FF AD	(00)	TO	DA	(80)
DEF	98	(80)	LOF	FF AE	(80)	TAN	FF BF	(80)
DSKI$	FF C1	(00)	MOTOR	B4	(80)	TERM	DB	(80)
DSKF	FF 9D	(80)	MERGE	B5	(80)	TIME$	FF 84	(00)
DATE$	FF 80	(00)	MOD	FD	(80)	USING	DC	(80)
DRAW	A2	(80)	MKI$	FF AF	(00)	USR	DD	(00)
ELSE	99	(D0)	MKS$	FF B0	(00)	VAL	FF C0	(80)
END	9A	(80)	MKD$	FF B1	(00)	VIEW	FF 85	(80)
ERASE	9B	(80)	MID$	FF 81	(00)	VARPTR	FF C3	(80)
EDIT	9C	(90)	MON	B6	(80)	WIDTH	DE	(80)
ERROR	9D	(80)	MAP	FF B2	(80)	WINDOW	FF 86	(80)
ERL	FF 9E	(90)	NEXT	B7	(80)	WAIT	DF	(80)
ERR	FF 9F	(80)	NAME	B8	(80)	WHILE	E0	(80)
EXP	FF A0	(80)	NEW	B9	(80)	WEND	E1	(80)
EOF	FF A1	(80)	NOT	BA	(80)	WRITE	E2	(80)
EQV	FB	(80)	OPEN	BB	(80)	WBYTE	ED	(80)
FOR	9E	(80)	OUT	BC	(80)	XOR	FA	(80)
FIELD	9F	(80)	ON	BD	(80)	+	F3	(00)
FILES	A0	(80)	OR	F9	(80)	-	F4	(00)
FN	A1	(00)	OCT$	FF B3	(00)	*	F5	(00)
FRE	FF C2	(80)	OPTION	BE	(80)	/	F6	(00)
FIX	FF A2	(80)	OFF	BF	(80)	^	F7	(00)

FPOS	FF A3	(80)	PRINT	C0	(80)	¥	FE	(00)
GOTO	A3	(90)	PUT	C1	(80)	>	F0	(00)
GO TO	A3	(90)	POKE	C2	(80)	=	F1	(00)
GOSUB	A4	(90)	POS	FF B4	(80)	<	F2	(00)
GET	A5	(80)	PEEK	FF B5	(80)	?	C0	(00)
HEX$	FF A4	(00)	PSET	C3	(80)			

表 2-3 中間コード表(　)の中はフラグ

中間コードを昇順に出力するプログラム(リスト2-1, 2-2)では，本来は，ソートプログラムが必要となる分けですが，配列の添字を使ってうまく自動的にソートするようになっています。これは，中間コードが，256 個以下と少ないこと，キーワードと 1 対 1 に対応していて二重になることがないこと，という性格があるからできたことなのです。詳しくは，プログラムを読んで下さい。

リスト2-3 キーワードの頭文字の昇順に表示するプログラムでは，キーワードが，頭文字の順に整理されて，テーブルを作っているので，ソートプログラムは必要ありません。

2-3 ラベルテーブル

ラベルテーブルは，シンボルテーブルセグメントに存在し，セグメント 60 H の，6 AEH，6 AFH で示されるアドレスから，6B0H，6B1H で示されるアドレスまでです。

ラベルは，この中に図 2-3-1 の様な形式で頭文字のアルファベット順に登録されます。

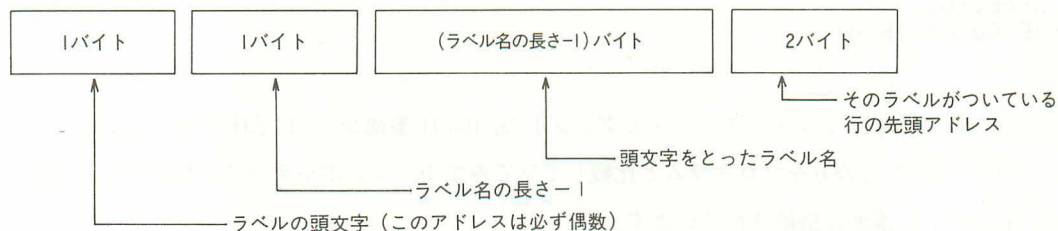



図2-3-1 ラベルテーブル中のラベルの登録

注意しなければならないのは，個々のラベルの情報が格納される先頭アドレスは，必ず偶数であるということです。そのために，もし，次のラベルが奇数アドレスで始まるようであれば，ラベル名のあとに無用の 1 バイトをもうけて，アドレスを調節しているのです。なぜこのようにしたかというと，8086 は奇数アドレスより偶数アドレスをアクセスする方が速いからなのです。

それでは，実際にラベルがどのような形で登録されるか見てみましょう。

次のプログラムを入力した後，RUN を実行して下さい。

```
1000 END
1010 *START
1020  GOSUB *INITIALIZE
1030 '
1040 *MAIN
1050  R=RND*500
1060  X=RND*600 : Y=RND*180
1070  CLR=INT(RND*7+1)
1080  CIRCLE(X,Y),R,CLR
1090 GOTO *MAIN
1100 '
1110 *INITIALIZE
1120  SCREEN 0,0
1130  RANDOMIZE
1140  CLS 3
1150 RETURN
```

最初の1000行にENDがあるので，この1000行だけを実行して，このプログラムは終了しますが，全てのラベルは登録されています。これは，RUN コマンドの最初にインタプリタ内で全ラベルを登録するルーチンをCALL しているためです。

これでラベルテーブルができあがりました。各ポインタの値をみてみましょう。

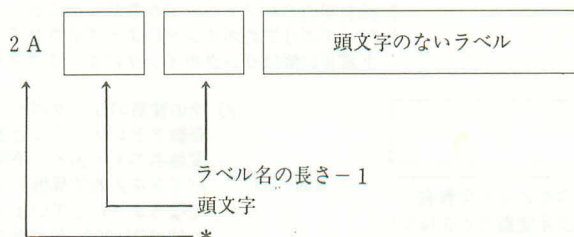
```
MON
h]C60
h]D6AE,6B1
06AE 00 01 1E 01
```

ラベルテーブルは，シンボルテーブルセグメントの，100 H 番地から，11 EH 番地にあるわけですね。それではそちらの方をプログラムと比較して見てみます。シンボルテーブルセグメントの値は，1410 H，1411 H 番地に格納されています。

```
h]D1410,1411
1410 00 16                 ; ROM版では0010即ち1000Hになります。
h]C1600                    ; モニタのアクセスするセグメントをかえます。
h]D0,5                     ; シンボルテーブルセグメントのワークエリアをみる。
0000 00 01 1E 01 1E 01     ; BASICのワークエリアの6AE～6B1と同じものがここにもある。
h]D100,11E
0100 49 09 4E 49 54 49 41 4C 49 5A 45 41 EE 23 4D 03   I NITIALIZEA/#M
0110 41 49 4E 54 78 23 53 04 54 41 52 54 4F 23 00      AINTx#S TARTO#
```

```
h]C60            ; BASICのワークエリアのセグメント
h]D6A4,6A7       ; BASICのテキストTOP, END
06A4 48 23 22 24

h]D2348,2422
2348 07 00 E8 03 01 9A 00 0D 00 F2 03 01 2A 53 04 54    ♠  ┗    年  *S T
2358 41 52 54 00 14 00 FC 03 02 A4 01 2A 49 09 4E 49   ART         、 *I NI
2368 54 49 41 4C 49 5A 45 00 08 00 06 04 01 00 27 00   TIALIZE           ´
2378 0C 00 10 04 01 2A 4D 03 41 49 4E 00 0F 00 1A 04         *M AIN
2388 02 52 00 F1 FF B7 F5 1C F4 01 00 1A 00 24 04 02    R 円 ｷ時 日      $
2398 58 00 F1 FF B7 F5 1C 58 02 01 3A 01 59 00 F1 FF   X 円 ｷ時 X   : Y 円
23A8 B7 F5 0F B4 00 15 00 2E 04 02 43 02 4C 52 F1 FF   ｷ時 ｴ    .  C LR円
23B8 A6 28 FF B7 F5 17 F3 11 29 00 16 00 38 04 02 8D   ｦ( ｷ時 月 )    8 █
23C8 28 58 00 2C 59 00 29 2C 52 00 2C 43 02 4C 52 00   (X ,Y ),R ,C LR
23D8 0E 00 42 04 01 A3 01 2A 4D 03 41 49 4E 00 08 00     B  ｣ *M AIN
23E8 4C 04 01 00 27 00 12 00 56 04 01 2A 49 09 4E 49   L   ´     V  *I NI
23F8 54 49 41 4C 49 5A 45 00 0B 00 60 04 02 D0 01 10   TIALIZE      `  ﾐ
2408 2C 10 00 07 00 6A 04 02 CE 00 09 00 74 04 02 8F   ,      j  ﾎ    t  +
2418 01 13 00 07 00 7E 04 01 C6 00 00                         ~  ﾆ
```

先程のサンプルプログラムの内部表現です。テキストの中ではラベルは，

という形でおかれています。ラベルの先頭の＊は，掛算の中間コードF5Hには変換されませんのですぐ区別することができます。

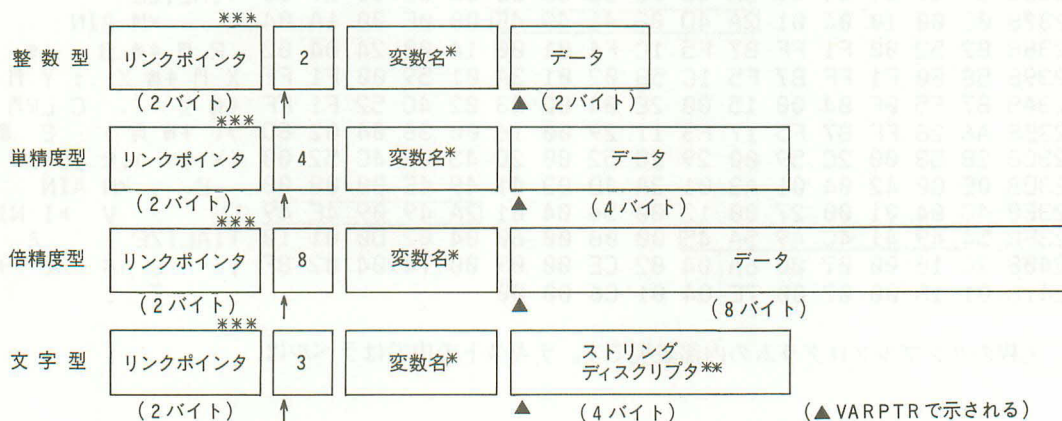
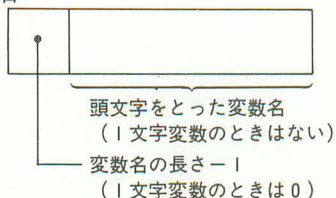
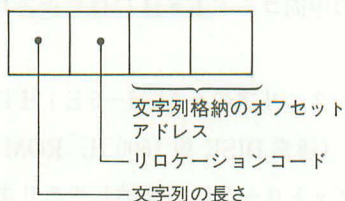
ラベルテーブルのオフセットアドレスはBASICのワークエリアの6AEH～6B1Hにあると述べましたが，同じものが，シンボルテーブルセグメント（通常DISK版1600H，ROM版1000H，これは1410H，1411Hに格納してあります）の，オフセット0～5にも格納してあります。

2-4　変数テーブル

2-4-1 単純変数テーブル

単純変数テーブルは，ラベルテーブルの後に作られます。この領域は，シンボルテーブルセグメントのオフセット，0002H，0003Hで示されるアドレスから，0004H，0005Hで示されるアドレスの1つ前までです。

プログラム中で変数が使われると，その型に応じた形式で，それが使われていた順番に登録されていきます。

PC-8001やPC-8801と違って，リスト形式を使って，頭文字のアルファベット順にリンクポインタをもって，並んでいます。各変数は次の様な形式で登録されます。

＊＊＊）リンクポインタ

各変数は頭文字を除いた形で並んでいますので，1文字変数のときは，数字データだけがあるという感じになります。これだけみても，どのデータがどの変数に対応しているのか分かりません。では，どのようにして，対応づけられているのでしょうか？

頭文字がAの最初の変数の位置は，シンボルテーブルセグメント（セグメント60Hの1410H,

1411 H に格納されています）のオフセットアドレス 0～FFH の 256 バイトにワークエリアがありますが，この 3 CH，3 DH に格納されているのです。あと，順に 3 EH，3 FH が B に，40 H，41 H が C に，…6 EH，6 FH が Z に対応しています。

そして，頭文字が A の 2 番目の変数は，1 番目の変数のリンクポインタによって，さされたところにあるのです。

これを図に示すと次のようになります。

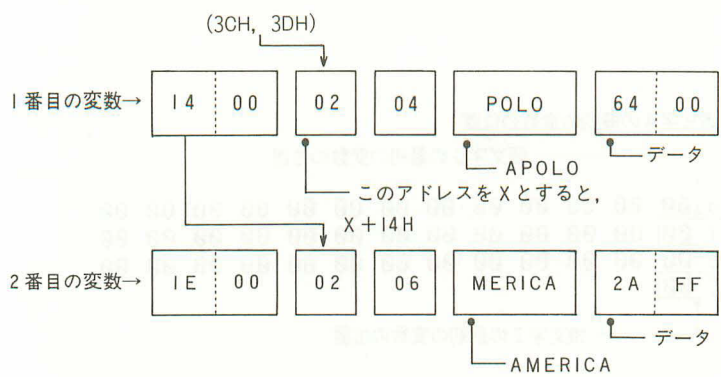



図 2-3-1 変数のリンクポインタ

実際の例で確かめてみましょう。

次のプログラムを実行した後，各ポインタの値及び単純変数テーブルを見てみましょう。

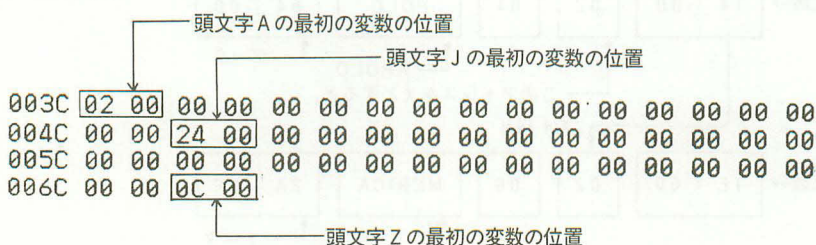
```
100 APOLO%=100
110 ZEBRA%=-100
120 AMERICA=500
130 JAPAN#=5#
140 ABC$="12345"
```

```
hJC1600        ; DISK 版の値（ROM 版は1000H です。）
hJD0,5
0000 00 01 00 01 3C 01
```

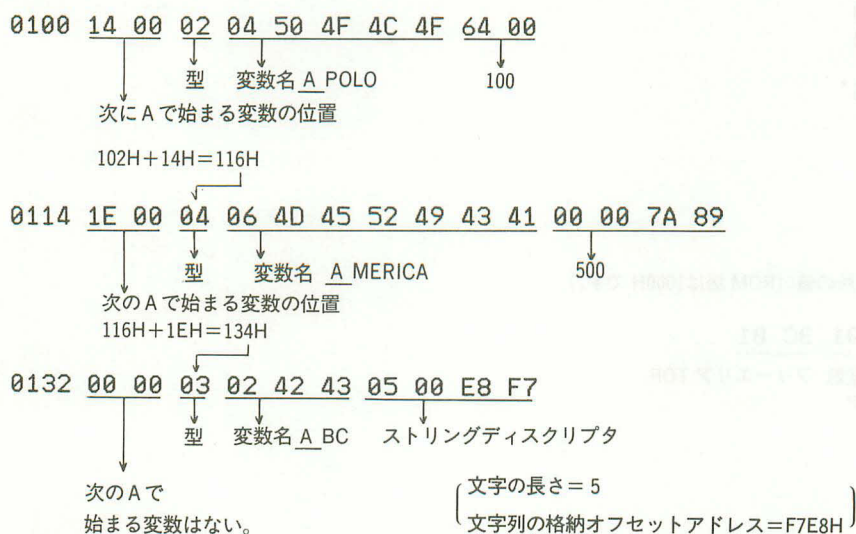
ラベルテーブル TOP　単純変数エリア TOP　フリーエリア TOP

※単純変数テーブル

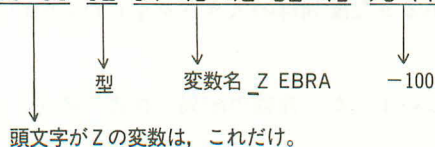
```
hJD100,13B
0100 14 00 02 04 50 4F 4C 4F 64 00 00 00 02 04 45 42   POLOd     EB
0110 52 41 9C FF 1E 00 04 06 4D 45 52 49 43 41 00 00 RA    MERICA
0120 7A 89 00 00 08 04 41 50 41 4E 00 00 00 00 00 00 z    APAN
0130 20 83 00 00 03 02 42 43 05 00 E8 F7                  BC
```

※最初の変数の頭文字のリンクテーブル

ちょっと分かりにくいので，各変数ごとに見ていきましょう。

● APOLO%＝100，AMERICA＝500，ABC$＝〝12345〞

オフセットアドレス 3 CH，3 DH をみると頭文字 A の最初の変数の位置は 0002 です。そこで単純変数 TOP のオフセットアドレス 100 H に 0002 H を足して，102 H 番地をみればよいことが分かります。

```
hJDF7E8
F7E8 31 32 33 34 35 05 00 00 44 32 2D 32 33 00 00 00
```

● ZEBRA%＝−100

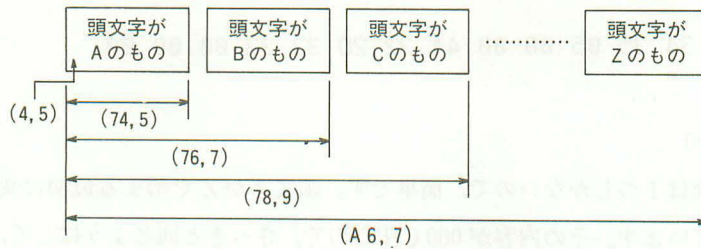
頭文字がZの変数は1つしかないので，簡単です。頭文字がZで始まる最初の変数の位置は，6 EH，6 FHに入っています。その内容が000 CHなので，さっきと同じようにして，10 CH番地をみればよいことが分かります。

● JAPAN#＝5#

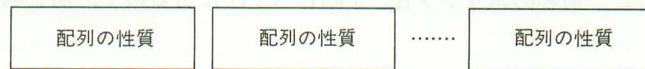
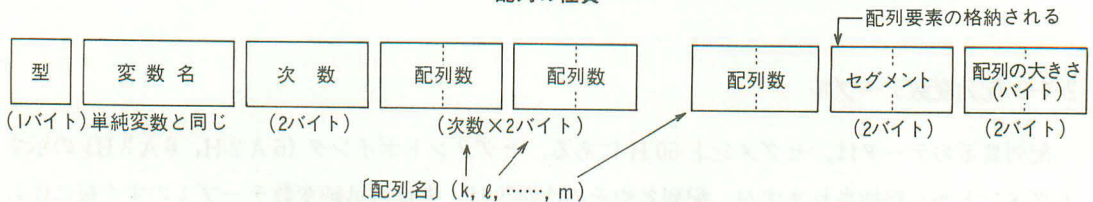
頭文字がJの場合は，最初のポインタが，4 EH，4 FHにあるので，前と同様にして，

2-4-2 配列変数テーブル

配列要素のデータは，セグメント60 Hにある，セグメントポインタ（6 A 2 H，6 A 3 H）の示すセグメントから格納されますが，配列名やその次数等は，先程の単純変数テーブルのすぐ後に作られます。

このテーブルのアドレスは，セグメントポインタ（1410 H，1411 H）の示すセグメント（通称シンボルテーブルセグメント）のワークエリア（0004 H，0005 H）に格納されています。

単純変数と同じように，頭文字のアルファベットで並べてありますが，前のように，リスト構造をとっていません。出てきた順ではなくて，次の図のようになっています。

ポインタ（74 H，75 H）には頭文字がAの配列群の大きさが入っています。(76 H，77 H）には，頭文字がBの配列群の大きさと，それまでに出てきた配列群の大きさを足したものが入っています。以下Zまで同様に続きます。

個々の配列群の中は，次のように，頭文字が等しい1つか，複数の配列の性質で構成されています。

各配列の性質とは，次のような構成をとります。

配列群にどのように格納されているか，実際に見てみましょう。

●整数型・単精度型

```
100 DIM ABC%(3,2),ASA(2),SYSTEM(1,1,1)
110  FOR I=0 TO 3
```

```
120   FOR J=0 TO 2
130     ABC%(I,J)=I*10+J
140   NEXT J
150 NEXT I
160 '
170 FOR I=0 TO 2
180   ASA(I)=I*100
190 NEXT I
```

※ポインタの値

```
hJC1600
hJD0
0000 00 01 00 01 10 01 00 00 00 00 00 00 00 00 00 00
```

ラベルテーブル TOP（00 01）　単純変数 TOP（00 01）　配列群 TOP（10 01）

```
hJD70,A7
```

```
0070 10 00 00 00 1A 00 1A 00 1A 00 1A 00 1A 00 1A 00
0080 1A 00 1A 00 1A 00 1A 00 1A 00 1A 00 1A 00 1A 00
0090 1A 00 1A 00 1A 00 1A 00 2E 00 2E 00 2E 00 2E 00
00A0 2E 00 2E 00 2E 00 2E 00
```


頭文字がBから，Rまでは，1AHになっていますが，これは，1つ前のポインタとの差をとると0となることを意味します。そういう配列はないということです。それでは，Aはどうかというと，1つ前が0000になっているので

001A−0000＝1A

ということになり，1AH（＝26）バイトのエリアが確保されています。そういう配列は，あるということです。

頭文字がTからZまでも等しく，2EHとなっていますが，頭文字がTからZの配列はないということを意味しています。

次の図に，配列群の格納のようすを示します。

この例では，頭文字がAのものが

ABC%，ASA

の２つあり，Ｓのものが，

SYSTEM

の１つあることに注意して見て下さい。

ここまでが頭文字がAの配列群です。(74,5)＝1A ですが、110～129までがちょうど1AH (＝26) バイトになります。

```
013A 02 26 20 00
```

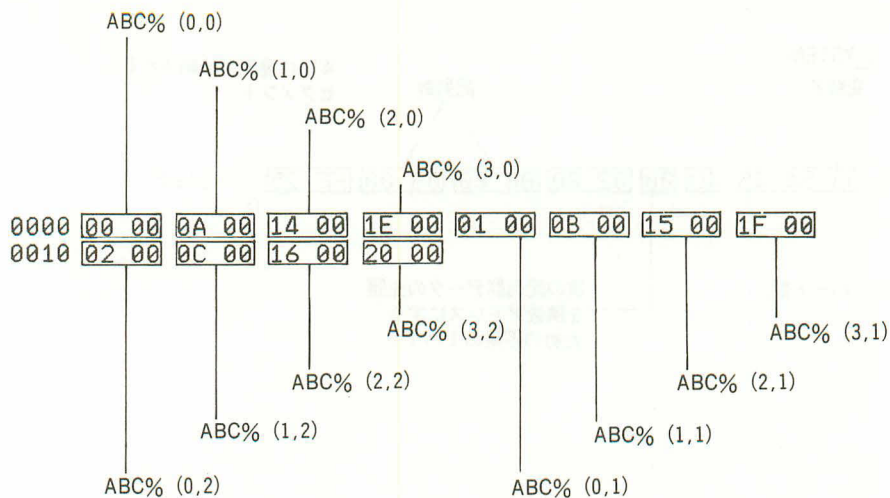
配列要素が格納されるセグメント　配列の大きさ(バイト数)

ここまでが頭文字がＳの配列群です。配列群Ｓのポインタ(98,9)＝2EH ですが、１つ前のポインタ (96,7)＝1AH の値をひいて、

２ＥＨ－１ＡＨ＝１４Ｈ（＝２０）

頭文字がＳの配列群の大きさは20バイトということになります。

実際に，配列要素が格納される場所は，上記テーブルで示されたアドレスです。配列 ABC%の要素をみてみましょう。

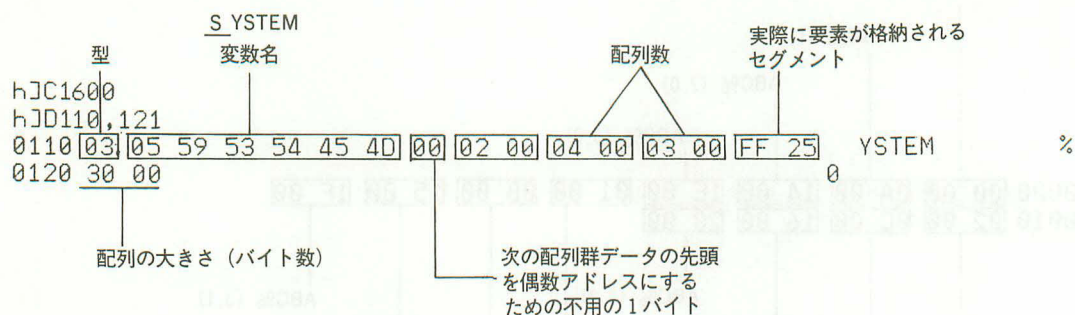
```
h]C25FF
h]D0,17
```

●文字型配列変数

```
10 DIM SYSTEM$(3,2)
20  FOR I=0 TO 3
30   FOR J=0 TO 2
40     SYSTEM$(I,J)=STR$(I*10+J)
50   NEXT J
60  NEXT I
```

※ポインタ値

```
hJC1600
hJD70,AF
0070 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0090 00 00 00 00 00 00 00 00 12 00 12 00 12 00 12 00
00A0 12 00 12 00 12 00 12 00 00 00 00 00 00 00 00 00
```

頭文字Sの配列のポインタ

12H バイト存在することを示している。

```
hJC25FF
hJD0,2F
```

ストリングディスクリプタ

```
0000 02 00 EB F7 03 00 E1 F7 03 00 D5 F7 03 00 C9 F7   ♣秒  ト秒  ユ秒  ノ秒
0010 02 00 E8 F7 03 00 DD F7 03 00 D1 F7 03 00 C5 F7   ♠秒  ン秒  ム秒  ナ秒
0020 02 00 E5 F7 03 00 D9 F7 03 00 CD F7 03 00 C1 F7   ◣秒  ル秒  ヘ秒  チ秒
```

(0, 0) (0, 1) (0, 2)	(1, 0) (1, 1) (1, 2)	(2, 0) (2, 1) (2, 2)	(3, 0) (3, 1) (3, 2)

に対応する。

参考までに，ストリングディスクリプタの指しているアドレスの中身を示しておきます。

```
hJC1600
hJDF7C1,F7EC
F7C1 20 33 32 03 20 33 31 03 20 33 30 03 20 32 32 03   32  31  30  22
F7D1 20 32 31 03 20 32 30 03 20 31 32 03 20 31 31 03   21  20  12  11
F7E1 20 31 30 03 20 32 02 20 31 02 20 30               10  2  1  0
```

2-5 文字列エリア

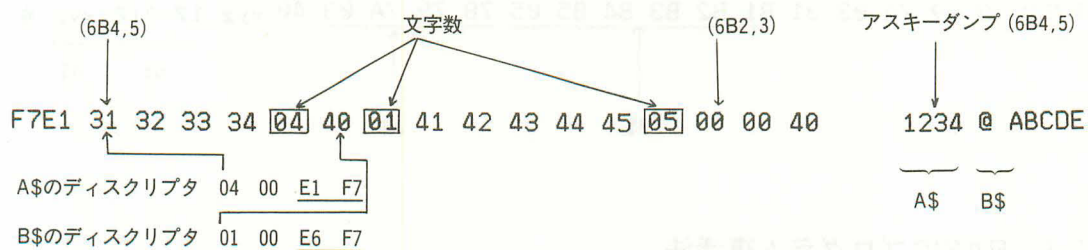
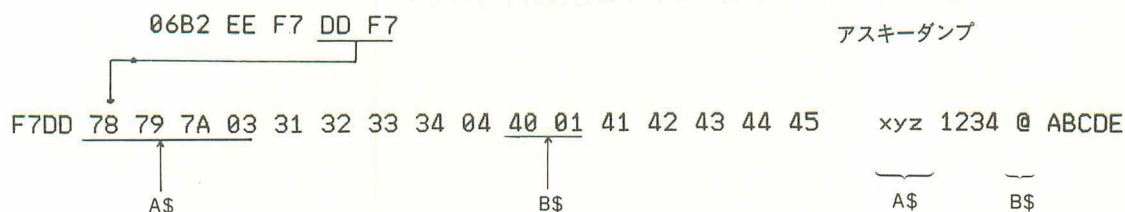
文字列エリアには文字型変数の実際の文字列が納められています。
例として次のプログラムを実行します。

```
10 A$="ABC"+"DE"
20 B$=CHR$(64)    ;CODE OF '@'
30 A$="1234"
```

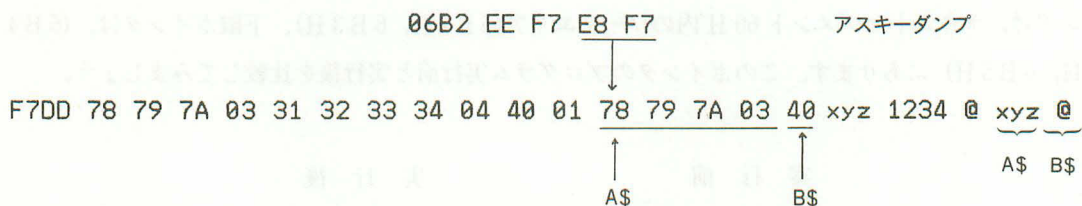
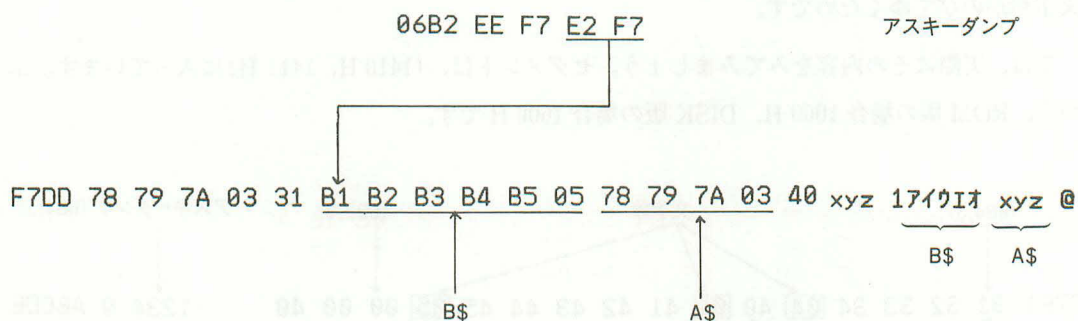
文字列エリアは，シンボルテーブルセグメントの上位アドレスに位置しています。その上限ポインタは，テキストセグメント60 H内のワークエリア(6 B 2 H，6 B 3 H)，下限ポインタは，(6 B 4 H，6 B 5 H) にあります。このポインタのプログラム実行前と実行後を比較してみましょう。

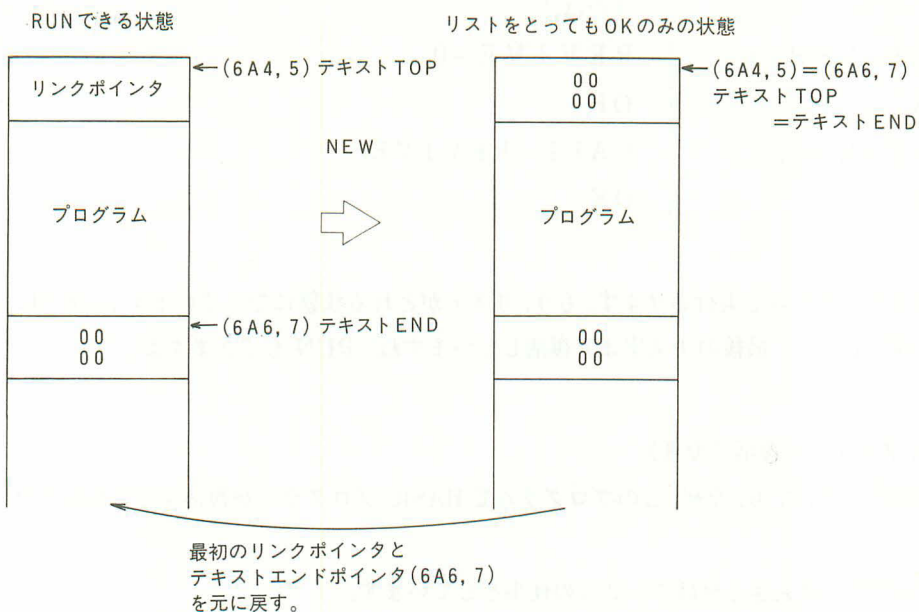
実行前		実行後
06B2 EE F7 EE F7	⇨	06B2 EE F7 E1 F7

ポインタが動きました。よくみると，(6 B 2 H，6 B 3 H) の方のポインタは動いていませんね。上限は固定されていて，文字列が増えると，下限のポインタが更新されて，アドレスの下位の方へ文字列がのびてゆくためです。

では，実際にその内容をみてみましょう。セグメントは，(1410 H，1411 H)に入っています。ふつう，ROM 版の場合 1000 H，DISK 版の場合 1600 H です。

最初に A$に代入された「ABCDE」はメモリ上から消えずにそのまま残っているのです。ここでさらにダイレクトモードで A$= "xyz" と行うと次のようになります。

ここで，ガベージコレクションを行ってみましょう。

? FRE(0)

こうすると次のようになります。

現在使用されている「xyz」と「@」だけが，文字列エリアの後からつめられて，(6B4H，6B5H)が移動しました。ここで，B$= "アイウ"+ "エオ" と行うと，次のようになります。

2-6 BASICプログラム復活法

リセットをしたり，電源を1度切ったりすると，テキスト・エリアは0クリアされてしまい復活は不可能ですが，NEWをした場合は，0クリアされずに，リンクポインタとプログラムの終わりを示すポインタがクリアされるだけで，プログラム本体は，まだメモリに残っています。したがってこれらの値を元に戻してやればプログラムは復活するのです。

これを自動的に行うプログラムを次に紹介します。

```
0000 FC 16 1E A1 A4 06 05 04 00 A3 EA 06 BF 47 00 CD
0010 C4 AC 08 C0 75 F6 8B 04 3D 05 01 72 07 46 46 AC
0020 08 C0 75 FB 8B 1E A4 06 29 DE 89 37 8B 07 09 C0
0030 74 04 01 C3 EB F6 89 1E A6 06 CF
```

BASICプログラム復活（セグメント1D00H，オフセット0H）

NEWをした直後に

```
CLEAR ,&H1D00
OK
DEF SEG=&H1D00
OK
MON
h] S0
0000  00-FC  00-……
```

というように、モニタを使って、上のプログラムを正確に打ち込みます。

```
                    または、
DEF USR=0        | REVIVE=0
OK               | OK
A=USR (0)        | CALL REVIVE
OK               | OK
```

として前記プログラムを実行させます。もう，リストがとれる状態になっていますよ。試みに，LIST をとってみて下さい。最後の1文字まで復活していますね。RUN もできますよ。

《BASIC プログラム復活の原理》

理屈は分かっていても，なぜ，このプログラムで BASIC プログラムが復活するのか知りたいところでしょう。

このプログラムは大きく分けて，2つの仕事をしています。

1つは，

●第1行のリンクポインタの値を計算する。

もう1つは，

●プログラム終了番地を計算する。

この2つです。

1つめのリンクポインタの方はちょっとやっかいで，

○テキスト TOP のアドレス

○文のよみとばし

○ REM 文の前の0と行の終わりを示す0との区別

という3大ポイントがあります。特に，REM 文の前にも0があるところがくせものです。インタプリタは，文中に0があると，リンクポインタを使って，次の行の先頭を割り出し，さっさと次の行を実行してしまいますが，我々は今，そのリンクポインタの値を知りたいわけですから困ったものです。テキストの中の REM をモニタで他の文字に変えても REM 文の役わりをします。それで，この区別をこうして行いました。

REM の方は，

```
        00 52 45 40
または，00 27 20
```

（20：ここは文字のコードになる。普通空白のコード以上です。(≧20 H)）

となっているはずですから，00のあとの2バイトをとってくると，

4552 H，又は2027 H以上

ということになります。

ところが，文の終わりの00は，次がリンクポインタですから，

```
0 0  ┌──────────┐
     │リンクポインタ│
     └──────────┘
      └────┬────┘
         2バイト
```

00のあとの2バイトをとると，リンクポインタそのものの値になります。リンクポインタの値は，

0000～0105H

程度のはず(1行は255文字以内なので) ですから，

105Hより大きかったら，REM文

〃　　小さかったら，行のおわり

と判断しました。

そして，REM文のよみとばしは，基本的にREM文の中にヌルコード00は入り込まないとして，行いました。

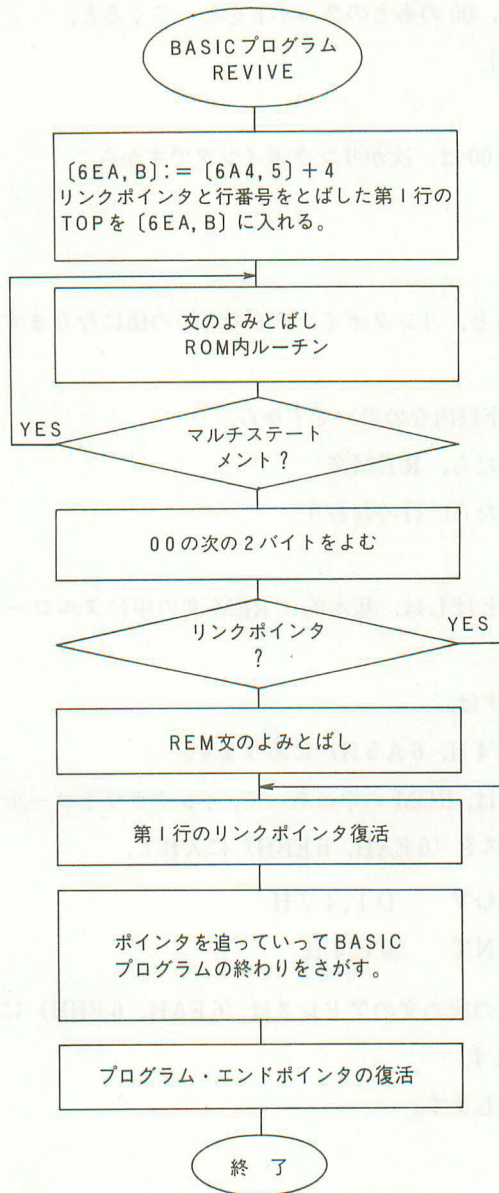
テキストTOPのポインタは，

セグメント60 Hの(6A4H，6A5H) にあります。

文のよみとばしルーチンは，ROMの中にあって，インタラプトコールで呼び出します。よみとばしたい部分のTOPアドレスを(6EAH，6EBH) に入れて，

```
BF 47 00        MOV    DI,47H
CD C4           INT    0C4H
```

とすればよいのです。出力の次の文のアドレスは(6EAH，6EBH) に入りますが，同じものが，SIレジスタにも入っています。

フローチャートを次に示します。

プログラムソースリストを次に示します。モニタのアセンブラ（DISK BASIC 起動時）で入力できます。

```
0000 FC        CLD                 ; LODS の方向を増加の方にする。
0001 16        PUSH    SS          ; DS の値を60H にする。
0002 1F        POP     DS
0003 A1A406    MOV     AX,[06A4]   ; テキスト TOP を6EA,B に入れる。
0006 050400    ADD     AX,0004
0009 A3EA06    MOV     [06EA],AX
```

```
000C BF4700        MOV    DI,0047     ；文のよみとばし。
000F CDC4          INT    C4
0011 AC            LODSB              ；マルチステートメントなら文のよみとばしをくり返
0012 08C0          OR     AL,AL       ；す。
0014 75F6          JNE    000C
0016 8B04          MOV    AX,[SI]     ；次がリンクポインタなら行のおわり。
0018 3D0501        CMP    AX,0105
001B 7207          JB     0024
001D 46            INC    SI          ；そうでなければREM文だ！
001E 46            INC    SI          ；行のおわりの0になるまで，LODSBをくりかえす。
001F AC            LODSB
0020 08C0          OR     AL,AL
0022 75FB          JNE    001F
0024 8B1EA406      MOV    BX,[06A4]   ；第1行の長さを計算する。
0028 29DE          SUB    SI,BX
002A 8937          MOV    [BX],SI     ；それを第1行のリンクポインタにしまう。
002C 8B07          MOV    AX,[BX]     ；リンクポインタをたどって，プログラムのおわりを
002E 09C0          OR     AX,AX       ；捜す。
0030 7404          JE     0036        ；リンクポインタが0なら，プログラムのおわり
0032 01C3          ADD    BX,AX       ；だ！
0034 EBF6          JMPS   002C
0036 891EA606      MOV    [06A6],BX   ；プログラムエンドポインタを復活する。
003A CF            IRET
```

BASIC復活プログラムソースリスト

注）BASICのワークエリアは，主に，セグメント60Hにありますが，この値は，モニタに入ったとき，又は，USRやCALLでマシン語プログラムをよんだときには，SSセグメントレジスタに入っています。

このプログラム中で，BASICのワークエリアを扱いますので，DSを60Hにする必要がありますが，

```
16    PUSH  SS
1F    POP   DS
```

とすれば，たった2バイトで，DSに60Hを入れることができます。

2-7 ステートメント・関数処理アドレス

PC-9801は，ROMのバージョンによって，ステートメント，関数の処理アドレスが異なります。ワークエリアにも，処理アドレスのテーブル先頭は格納されていません。しかし，ROMの中には，その処理アドレステーブルの先頭アドレスのベクタがあります。

以下に，処理アドレスの表を作成するプログラムを示します。実行サンプルを示しておきますが，ROMによって値が異なることがあるので，マシン語プログラム等作製する場合は，直接にこの処理アドレスを用いるようなことは，絶対にしないで下さい。（表は，オフセットの値です。セグメント

はE 800 Hです。)

関数の処理アドレスは,

アドレス1:アドレス2

となっていますが,アドレス1が,関数の引数を抽出するような前処理,アドレス2がその関数の核の処理となっています。

DATE$, MID$, POINT,

PEN, TIME$, VIEW

WINDOW

の7つは,ステートメント表にも関数表にも出てきますが,例えば,

関数として　　　A$=MID$ (A$,3,1)

ステートメントとして　MID$ (A$,3,1)= "THIS"

のように,2つの使い方があるからです。

プリンタに出力する場合は,100行のファイルディスクリプタの

"SCRN:"

を

"LPT1:"

にかえて下さい。

ステートメントの処理アドレス表を作成するプログラム

```
0 'SAVE"KEYADD.ST"
100   OPEN "SCRN:" AS #1
110  DIM SN(10),CODE(144),KY.WORD$(144),FLG(144)
120   DEF FNWPK(X)=PEEK(X)+PEEK(X+1)*256
130   DEF FNPK$(X)=FNHX$(FNWPK(X))
140   DEF FNHX$(X)=RIGHT$("000"+HEX$(X),4)
150   DEF FNWD$(X$)=LEFT$(X$+SPACE$(10),10)
160   DEF FNKY$(X)=FNWD$(KY.WORD$(I+X))+" : "+FNHX$(PEEK(AD+(I+X)*2)+
                   PEEK(AD+(I+X)*2+1)*256)+SPACE$(6)
170 '
180  DEF SEG=&H60
190 '
200  AD=FNWPK(&H140A)                          :'GET KEY WORD TABLE POINTER
210  BSE=FNWPK(&H140C)                         :' GET SEGMENT
220  CHR=ASC("A"):DEF SEG=BSE
230  WHILE CHR<92
240   ADD=FNWPK(AD)                            :' NOW  ALPHABET ADDRESS
250   NADD=FNWPK(AD+2)                         :' NEXT ALPHABET ADDRESS
260    GOSUB *LST                              :' GET KEY WORD LIST
270   AD=AD+2:CHR=CHR+1                        :' NEXT ALPHABET
280  WEND
290 GOTO *PRT.OUT                              :' PRINT OUT KEY WORDS
300 '
```

```
310 *LST
320  WHILE ADD<NADD
330   LN=PEEK(ADD)
340    FOR I=1 TO LN:SN(I)=PEEK(ADD+I):NEXT I
350     CODE=PEEK(ADD+LN+1):IF CODE<128 THEN 410
360     CNT=CODE-128
370     IF CHR=ASC("[") THEN KY.WORD$(CNT)="" ELSE KY.WORD$(CNT)=CHR$(CHR)
380      FOR I=1 TO LN
390       KY.WORD$(CNT)=KY.WORD$(CNT)+CHR$(SN(I))
400      NEXT
410    ADD=ADD+LN+3
420  WEND
430 RETURN
440 '
450 *PRT.OUT
460 '
470  DEF SEG=0:FL=0
480   O1=PEEK(&H310)+PEEK(&H311)*256
490   S1=PEEK(&H312)+PEEK(&H313)*256
500  DEF SEG=S1
510   O2=PEEK(O1+7)+PEEK(O1+8)*256
520   O3=PEEK(O2+54)+PEEK(O2+55)*256
530    FOR I=O3 TO O3+256              : REM FIND 'CMP AL,0FFH JZ'
532     IF PEEK(I)=&H3C AND PEEK(I+1)=&HFF AND PEEK(I+2)=&H74 THEN II=I+4
534    NEXT
536    FOR I=II TO II+256
540     IF PEEK(I)=&H8D AND PEEK(I+1)=&H3E THEN O4=I+2:GOTO 570
550     IF PEEK(I)=&HBF THEN O4=I+1:GOTO 570
560    NEXT
570  AD=PEEK(O4)+PEEK(O4+1)*256
580 '
590  FOR I=0 TO 47
600   PRINT #1,FNKY$(0);
610   IF I+&HB0<256 THEN PRINT #1,FNKY$(&H30);
620   IF I+&HE0<254 THEN PRINT #1,FNKY$(&H60) ELSE PRINT #1,""
630  NEXT
640 '
650  PRINT #1,"":PRINT #1," *** FUNCTION AS STATEMENT ***"
660  PRINT #1,"DATE$   : ";FNPK$(AD+252)
670  PRINT #1,"MID$    : ";FNPK$(AD+254)
680  PRINT #1,"POINT   : ";FNPK$(AD+256)
690  PRINT #1,"PEN     : ";FNPK$(AD+258)
700  PRINT #1,"TIME$   : ";FNPK$(AD+260)
710  PRINT #1,"VIEW    : ";FNPK$(AD+262)
720  PRINT #1,"WINDOW  : ";FNPK$(AD+264)
730 '
740 END
```

関数の処理アドレス表を作成するプログラム

```
0 'SAVE"KEYADD.FN"
100  OPEN "SCRN:" AS #1
110  DIM SN(10),KY.WORD$(144)
120  DEF FNWPK(X)=PEEK(X)+PEEK(X+1)*256
130  DEF FNHX$(X)=RIGHT$("000"+HEX$(X),4)
```

```
140  DEF FNWD$(X$)=LEFT$(X$+SPACE$(10),10)
150  DEF FNPK$(X)=FNHX$(FNWPK(X))
160  DEF FNKY$(X)=FNWD$(KY.WORD$(I+X))+" : "+FNPK$(AD+(I+X)*4)
     +":"+FNPK$(AD+(I+X)*4+2)+SPACE$(3)
170  '
180  DEF SEG=&H60
190  '
200  AD=FNWPK(&H140A)                 :'GET KEY WORD TABLE POINTER
210  BSE=FNWPK(&H140C)                :' GET SEGMENT
220  CHR=ASC("A"):DEF SEG=BSE         :' SET THE SEGMENT
230  WHILE CHR<92
240   ADD=FNWPK(AD)                   :' NOW  ALPHABET ADDRESS
250   NADD=FNWPK(AD+2)                :' NEXT ALPHABET ADDRESS
260    GOSUB *LST                     :' GET KEY WORD LIST
270   AD=AD+2:CHR=CHR+1               :' NEXT ALPHABET
280  WEND
290 GOTO *PRT.OUT                     :' PRINT OUT KEY WORDS
300 '-----------------------------
310 *LST
320  WHILE ADD<NADD
330   LN=PEEK(ADD)
340    FOR I=1 TO LN:SN(I)=PEEK(ADD+I):NEXT I
350     CODE=PEEK(ADD+LN+1):IF CODE>127 THEN 410
360     CNT=CODE
370     IF CHR=ASC("[") THEN KY.WORD$(CNT)=""
                         ELSE KY.WORD$(CNT)=CHR$(CHR)
380      FOR I=1 TO LN
390       KY.WORD$(CNT)=KY.WORD$(CNT)+CHR$(SN(I))
400      NEXT
410     ADD=ADD+LN+3
420  WEND
430 RETURN
440 '
450 '
460 *PRT.OUT
470 '
480  DEF SEG=0
490   O1=FNWPK(&H310):S1=FNWPK(&H312)
500  DEF SEG=S1
510   O2=FNWPK(O1+7):O3=O2+28
520   O4=FNWPK(O3)+&H40:O5=FNWPK(O4)+1
530   O6=FNWPK(O5):IF O6>32768! THEN O7=O6-65536!
540   AD=FNWPK(O5+2+O7+&H12)
550    FOR I=0 TO 47
560     PRINT #1,FNKY$(0);
570     IF I<&H20 THEN PRINT #1,FNKY$(&H30) ELSE PRINT #1,""
580    NEXT
590
600 END
```

ステートメントの処理アドレス表 セグメント：E800H

Statement	Address	Statement	Address	Statement	Address
AUTO	2290	LINE	305F	WHILE	74DF
BSAVE	8BCC	LOAD	8F38	WEND	7510
BLOAD	8EE0	LSET	724D	WRITE	9A45
BEEP	50F2	LFILES	6B66	LIST	7735
CONSOLE	5244	MOTOR	511F	SEG	3D5D
COPY	70A0	MERGE	8E96	SET	46DE
CLOSE	51DB	MON	2A64	KINPUT	9A25
CONT	22FC	NEXT	7451	SRQ	172D
CLEAR	2333	NAME	6A5C	CMD	172D
CALL	9B2A	NEW	2A37	IRESET	172D
COMMON	9B7E	NOT	3D5D	ISET	172D
CHAIN	9B84	OPEN	515D	POLL	172D
COM	251E	OUT	710F	RBYTE	172D
CIRCLE	2D32	ON	269B	WBYTE	172D
COLOR	2F78	OPTION	51FD	KPLOAD	3E62
CLS	3022	OFF	3D5D		1728
DELETE	2476	?	06C5	>	3D5D
DATA	22F5	PUT	350E	=	3D5D
DIM	9C6C	POKE	4974	<	3D5D
DEFSTR	9C31	PSET	314C	+	3D5D
DEFINT	9C29	PRESET	3150	-	3D5D
DEFSNG	9C2D	PAINT	3181	*	3D5D
DEFDBL	9C25	RETURN	27AD	/	3D5D
DSKO$	4732	READ	9ADB	^	3D5D
DEF	72CE	RUN	28D5	AND	3D5D
ELSE	12C8	RESTORE	9B12	OR	3D5D
END	2A26		3D5D	XOR	3D5D
ERASE	9C88	RESUME	296B	EQV	3D5D
EDIT	2A75	RSET	727F	IMP	3D5D
ERROR	2324	RENUM	2B6A	MOD	1728
FOR	733A	RANDOMIZE	72AF		
FIELD	76B6	ROLL	3242		
FILES	6B6E	SCREEN	32C0		
FN	3D5D	STOP	16FC		
DRAW	3E5C	SWAP	71AD		
GO TO	2716	SAVE	8C27		
GOSUB	271C	SPC	3D5D		
GET	34A7	STEP	3D5D		
HELP	2548	THEN	3D5D		
INPUT	989A	TRON	76A6		
IF	2AE0	TROFF	76AD		
KEY	2596	TAB	3D5D		
KILL	6ACF	TO	3D5D		
KANJI	3D5D	TERM	6CD2		
LOCATE	536C	USING	3D5D		
L?	06BE	USR	3D5D		
LLIST	773C	WIDTH	54D9		
LET	719C	WAIT	70DF		

*** FUNCTION AS STATEMENT ***

Function	Address
DATE$	1728
MID$	5405
POINT	71FB
PEN	3505
TIME$	263D
VIEW	2A0C
WINDOW	3354

注）ROMによって値が異なりますので，先程のプログラムを入力し，ご自分のマシンのアドレス表を作って下さい。

関数の処理アドレス表 セグメント：E800H

DATE$	: ABF7:63C9	MKS$	: AC06:5FF4
MID$	: AC67:5ED2	MKD$	: AC06:6002
POINT	: ABF7:499C	MAP	: ABF7:487F
PEN	: ABF7:4B41	OCT$	: AC06:5EC0
TIME$	: ABF7:6403	POS	: ABF7:62E2
VIEW	: ABF7:4A53	PEEK	: ABF7:4957
WINDOW	: ABF7:4A68	RIGHT$	: AC49:5EFD
	: 0000:0000	RND	: ABF7:7558
	: 0000:0000	SEARCH	: ABF7:6440
	: 0000:0000	SGN	: AC06:4F9D
	: 0000:0000	SQR	: AC06:4F7C
	: 0000:0000	SIN	: AC06:4C32
	: 0000:0000	STR$	: AC06:5FA4
	: 0000:0000	STRING$	: AC97:5F1F
	: 0000:0000	SPACE$	: ABFD:5F16
	: 0000:0000	TAN	: AC06:4BF5
ABS	: AC06:4BB4	VAL	: AC0F:5F89
ATN	: AC06:4BBD	DSKI$	: ABF7:4726
ASC	: AC0F:5E8C	FRE	: ABF7:4AB2
ATTR$	: ABF7:46D2	VARPTR	: ABF7:9CA1
CSRLIN	: ABF7:62DC	INPUT$	: ABF7:99E0
CINT	: AC06:5BEE	JIS$	: ABF7:3E20
CSNG	: AC06:5C4D	KNJ$	: ABF7:3E26
CDBL	: AC06:5C66	KTYPE	: ABF7:3E2C
CVI	: AC0F:5FB1	KLEN	: ABF7:3E32
CVS	: AC0F:5FBD	KMID$	: ABF7:3E38
CVD	: AC0F:5FCF	KEXT$	: ABF7:3E3E
COS	: AC06:4C21	KINSTR	: ABF7:3E44
CHR$	: ABFD:5EA5	AKCNV$	: ABF7:3E4A
DSKF	: ABF7:46E4	KACNV$	: ABF7:3E50
ERL	: ABF7:7552	IEEE	: 7FBB:2000
ERR	: ABF7:7546	STATUS	: D1D3:D1E3
EXP	: AC06:4E4D		
EOF	: ABF7:4738		
FIX	: AC06:4CB8		
FPOS	: ABF7:4799		
HEX$	: AC06:5EB1		
INSTR	: AC18:5F42		
INT	: AC06:4CD0		
INP	: ABF7:7120		
INKEY$	: ABF7:600A		
LPOS	: ABF7:4806		
LOG	: AC06:4CF5		
LOC	: ABF7:4815		
LEN	: AC0F:5E9D		
LEFT$	: AC49:5ECF		
LOF	: ABF7:484B		
MKI$	: AC06:5FEF		

注）アドレスは、前処理：核処理のように表示されています。

第３章　テキスト画面

3-1　WIDTH

3-1-1　WIDTHとDIPスイッチ
3-1-2　WIDTH文のパラメータ省略

3-2　テキストVRAMのアドレス

3-3　画面とアドレスの対応表

3-4　アトリビュートエリア

3-5　テキスト画面でグラフィックが使える！

3-6　画面を縦に2分割

3-7　テキスト画面の2ページ目を利用

3-8　ひらがなの表示

3-9　TABキーとTAB関数

第3章　テキスト画面

3-1　WIDTH

3-1-1 WIDTH と DIP スイッチ

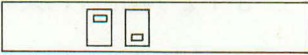
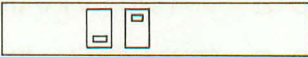
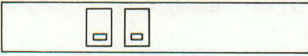
テキスト画面のモードには次の4つの組み合わせがあり，WIDTH 文や DIP スイッチ（本体後部にある SW 2）によって設定されます（図 3-1-1）。DIP スイッチによって設定されたモードは，電源投入時または，リセットボタンを押したとき有効となります。

画面モード	WIDTH文	DIPスイッチ(SW2)
40文字×20行	WIDTH　40,20	
40文字×25行	WIDTH　40,25	
80文字×20行	WIDTH　80,20	
80文字×25行	WIDTH　80,25	

図 3-1-1 DIP スイッチと WIDTH

3-1-2 WIDTH 文のパラメータの省略

WIDTH 文は，桁数と行数の2つのパラメータをもちます。N-BASIC での WIDTH 文はいずれも省略できますが（例えば，WIDTH,25，WIDTH,），N$_{88}$-BASIC (86) では，行数の省略しかできません。したがって，桁数がわかっていないときに行数だけを変えたい場合は次のようにします。

```
10 DEF SEG=&H60
20 WIDTH PEEK(&H460)+1,25'(または20)
```

この 460 H というのは画面の桁数－1がはいっているアドレスです。ちなみに，行数－1の値は 484 H 番地に入っています。

ただしこれは，BASIC インタプリタが画面モードの値を参照するためのものであって，これらのアドレスに値を POKE したからといって画面モードは変化しません（他に CRTC のモード設定が必要です）。

画面の大きさは，1バイト（≦255）で表わせますが，ワークエリアでは，2バイト用いています。

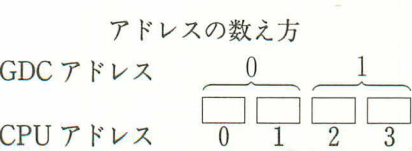
3-2 テキストVRAMのアドレス

テキストVRAMは，GDCμPD 7220（Graphic Display Controller）というLSIによってコントロールされています。CPUからみたVRAMアドレスは，１バイト単位ですが，GDCからみたVRAMアドレスは，２バイト（ワード）単位となっています。VRAMは，主記憶空間に配置されていて，CPUアドレスで，物理アドレス

A 0000 H～A 3 FFFH（画面＋アトリビュートエリア２枚分）

です。次に，GDCからみたアドレスとCPUからみたアドレスの対応を表3-2-1に示します。

通常は偶数アドレスのみを使用し，漢字オプション実装時，漢字表示のために奇数アドレスも使用します。

GDCアドレス（ワード単位）	CPUアドレス（バイト単位）	メモリイメージ	
		F E D C B A 9 8	7 6 5 4 3 2 1 0
００００	Ａ００００	日本語 テキスト表示 １ページ	ＡＮＫ／日本語 テキスト表示 １ページ
０８００	Ａ１０００	日本語 テキスト表示 ２ページ	ＡＮＫ／日本語 テキスト表示 ２ページ
１０００	Ａ２０００		アトリビュート １ページ
１８００	Ａ３０００		アトリビュート ２ページ

ただし、CPUアドレスのA4000～A7FFFの部分に拡張バスからメモリ等を増設することはできません。

アトリビュートの奇数アドレスには、メモリはありません。

表3-2-1 テキストVRAMのアドレス

3-3 画面とアドレスの対応表

テキスト画面に表示される文字は，VRAM（Video RAM）と呼ばれるN_{88}-BASIC(86) ワークエリア上のデータです。画面の１文字は，VRAMの２バイトに対応し，このVRAMにデータを書き込むことにより，画面に文字を表示することができます。また，このVRAMのデータを読むことで画面に表示されている文字コードを知ることもできます。

このVRAMには，各行について，320バイト（160バイトが文字コード，160バイトがアトリビュートコード）ずつ，合計8000バイトが２組で16000バイト使われています。

実際のVRAMのアドレスは次の表の通りです。表示されているのはオフセットですから，次のセ

グメントを加えて下さい。

テキスト画面１：A000H　　　アトリビュート１：A200H
テキスト画面２：A100H　　　アトリビュート２：A300H

つまり，テキスト画面１だと，

ＤＥＦ　ＳＥＧ＝＆ＨＡ０００

として下さい。

```
Line   Character     Attribute
  0   0000...009F   0000...009F
  1   00A0...013F   00A0...013F
  2   0140...01DF   0140...01DF
  3   01E0...027F   01E0...027F
  4   0280...031F   0280...031F
  5   0320...03BF   0320...03BF
  6   03C0...045F   03C0...045F
  7   0460...04FF   0460...04FF
  8   0500...059F   0500...059F
  9   05A0...063F   05A0...063F
 10   0640...06DF   0640...06DF
 11   06E0...077F   06E0...077F
 12   0780...081F   0780...081F
 13   0820...08BF   0820...08BF
 14   08C0...095F   08C0...095F
 15   0960...09FF   0960...09FF
 16   0A00...0A9F   0A00...0A9F
 17   0AA0...0B3F   0AA0...0B3F
 18   0B40...0BDF   0B40...0BDF
 19   0BE0...0C7F   0BE0...0C7F
 20   0C80...0D1F   0C80...0D1F
 21   0D20...0DBF   0D20...0DBF
 22   0DC0...0E5F   0DC0...0E5F
 23   0E60...0EFF   0E60...0EFF
 24   0F00...0F9F   0F00...0F9F
```

次に，カーソル位置からVRAMのアドレスを求める方法を示します。

※BASICによる方法

● 40ケタモード

V.ADRS=160＊CUR.Y+4＊CUR.X

● 80ケタモード

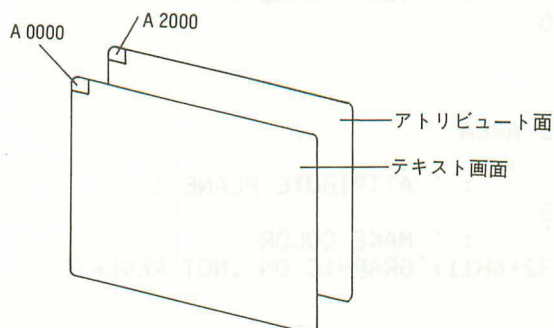
V.ADRS=160＊CUR.Y+2＊CUR.X

これらは，関数として定義しておくと便利です。

3-4 アトリビュートエリア

文字のブリンキングや色，下線などの制御を行うために，その属性（アトリビュート）を記録す

る部分が画面の文字と，1対1対応で存在します。

1文字は2バイトで表わされますが，対応するアトリビュートも2バイトです。ただし，上位8ビットは使用されません。アトリビュートの内容を図3-4-1に示します。

対応するアトリビュートの属性で表示が変化します。

7	6	5	4	3	2	1	0
緑	赤	青	垂線 簡易グラフ ＝1　注)	アンダーライン ＝1	リバース ＝1	ブリンキング ＝1	シークレット ＝0

カラーCRTの場合

0 0 0	黒	COLOR0	1 0 0	緑	COLOR4
0 0 1	青	COLOR1	1 0 1	水色(シアン)	COLOR5
0 1 0	赤	COLOR2	1 1 0	黄	COLOR6
0 1 1	紫(マゼンタ)	COLOR3	1 1 1	白	COLOR7

モノクロCRTの場合　● 明 7 ←——→ 0 暗　　濃淡になる。

注) I/Oポート68Hに、1……簡易グラフ，0……垂線をアウトするとよい。
シークレットは負論理で，0のときにシークレット表示し，1のときにノーマル表示します。

図3-4-1 アトリビュートエリア

3-5 テキスト画面でグラフィックが使える!

ビット4の簡易グラフというのは，PC-8001の160×100ドットグラフィックと同じものです。ただし，カラーは1文字分（2×4ドット）ずつですが，変化数については1行20回までというような制限はありません。次に，そのデモを示します。

簡易グラフィックデモンストレーション

```
0 ′SAVE"PC8001.GRP"
100 ′
110 ′   PC-8001 160*100 GRAPHIC DEMO
120 ′
130  DEF SEG=&HA000       : ′ TEXT PLANE 1
140   FOR I=0 TO 255
150    POKE I*2,I
160   NEXT
170 ′
180 ′   SET ATTRIBUTE AREA
190 ′
200  DEF SEG=&HA200       : ′ ATTRIBUTE PLANE 1
210   FOR I=0 TO 255
220    CL=I MOD 8         : ′ MAKE COLOR
230    POKE I*2,CL*32+&H11:′GRAPHIC ON ,NOT REVERSE
240   NEXT
250 END
```

ビット4には，垂線とも書いてありますがこの区別は，I/Oポート68Hで行います。

上の簡易グラフィックDEMOをRUNして，

OUT &H68,0

OK

として下さい。文字の中央に垂線が表示されたでしょう。

OUT &H68,1

として下さい。簡易グラフィックにもどったでしょう。

PC-8001の160×100ドットグラフィックをご存知ない方のために，その使い方を図3-4-2に示しておきます。

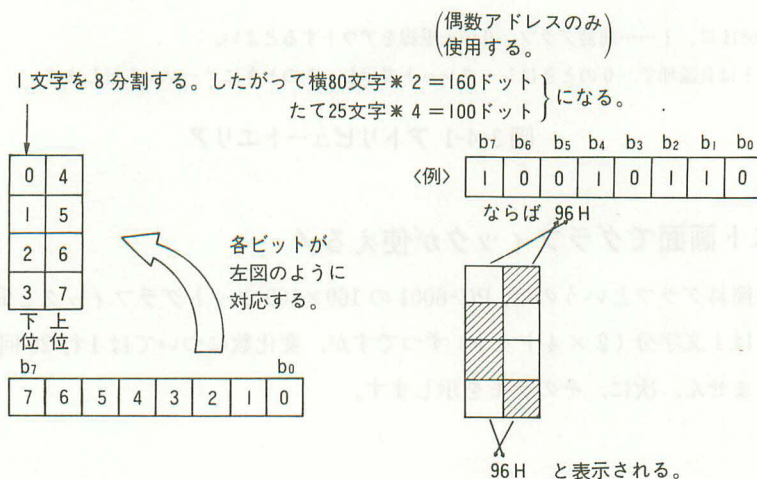



図3-4-2 160×100ドットグラフィック

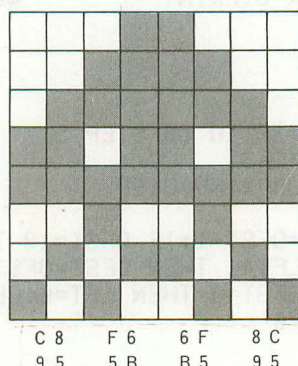
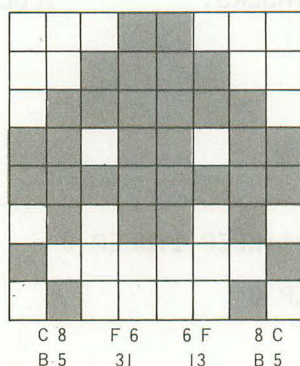
ランダムパターンでは面白くないので，

おなじみのインベーダーを表示するプログラムを紹介しましょう。インベーダーが左右に行ったり来たりします。STOP キーで，プログラムを中断することができます。

モノクロモードなのに，アトリビュートを直接いじることによって色をつけることもできますね。インベーダーの形は，文字を消すのと同じ方法（HOME CLR キーを押すなど）で消すことができます。

インベーダー表示プログラム

```
0 'SAVE"INVDEMO"
100 '
110 '  INVADER
120 '
130  WIDTH 80,25:CONSOLE ,,0:COUNT=-1
140  DIM INV1(11),INV2(11)
150 '
160  RESTORE *INV.PATTERN
170  FOR I=0 TO 11:READ A$:INV1(I)=VAL("&H"+A$):NEXT
180  FOR I=0 TO 11:READ A$:INV2(I)=VAL("&H"+A$):NEXT
190 '
200 '  SET ATTRIBUTE AREA
210 '
220  DEF SEG=&HA000
230   LOCATE 20,3:PRINT "Now set attribute area."
240    Y=8:ADD=Y*160
250      FOR X=0 TO 159
260       POKE ADD+X,0:POKE ADD+X+160,0
270      NEXT
280      FOR X=0 TO 74
290       GOSUB *ATR.SET
300      NEXT
310   LOCATE 13,3
320    COLOR 6:PRINT "       We are INVADERS.        ":COLOR 0
330 '
340 '  MAIN
350 '
360  Y=8
370   FOR J=10 TO 60 STEP 5
380     X=J+OFST
390     GOSUB *INV.DISP
400   NEXT
410    OFST=OFST+1:IF OFST=10 THEN LFT=1:BEEP 1:BEEP 0
420     IF LFT=1 THEN OFST=OFST-2
430     IF OFST=0 THEN LFT=0:BEEP 1:BEEP 0
440  GOTO 370
450 '
460 END
470 *INV.PATTERN
480  DATA 00,C8,F6,6F,8C,00
490  DATA 00,B5,31,13,5B,00
500 '
510  DATA 00,C8,F6,6F,8C,00
520  DATA 00,59,5B,B5,95,00
530 '
540 '  DISPLAY INVADER PATTERN
550 '
```

```
560 *INV.DISP
570  DEF SEG=&HA000
580   ADD=Y*160+X*2
590    COUNT=-COUNT:IF COUNT=-1 THEN 680
600     FOR I=0 TO 5
610       POKE ADD+I*2,INV1(I)
620     NEXT
630    ADD=ADD+160
640     FOR I=6 TO 11
650       POKE ADD+(I-6)*2,INV1(I)
660     NEXT
670  RETURN
680 '
690     FOR I=0 TO 5
700       POKE ADD+I*2,INV2(I)
710     NEXT
720    ADD=ADD+160
730     FOR I=6 TO 11
740       POKE ADD+(I-6)*2,INV2(I)
750     NEXT
760   RETURN
770 '
780 '    SET ATTIBUTE AREA SUBROUTINE
790 '
800 *ATR.SET
810  DEF SEG=&HA200
820   ADD=Y*160+X*2
830     FOR I=0 TO 5
840       POKE ADD+I*2,&H31
850     NEXT
860    ADD=ADD+160
870     FOR I=6 TO 11
880       POKE ADD+(I-6)*2,&H31
890     NEXT
900  RETURN
```

インベーダーパターンデータの作成法

3-6 画面を縦に2分割

画面を横に分けるには，図3-6-1のようにCONSOLEを使えばできます。

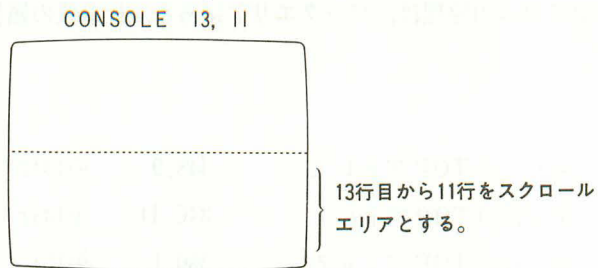



図 3-6-1 画面を横に分割

これから示すプログラムを使うと次の図 3-6-2 のように画面を縦に割って使うことができます。

図 3-6-2 画面を縦に分割

次のプログラムを実行して，LIST をとってみて下さい。画面の右側 40 文字分だけを使っているでしょう。

画面縦分割プログラム

```
0 'SAVE"FIRST.CHR"
100 '
110 '    MAKE HALF SIDE OF SCREEN
120 '   Copy rihght by T.K SYSTEMSOFT(C)
130 '
140  WIDTH 80,25
150  DEF SEG=&H60
160   FOR I=0 TO 24   : ' WHILE 0<=LINE<=24
170     X=&H314+I*4   : ' LINE TOP BUFFER
180     Y=160*I+40*2  : ' USE 2 BYTES BY 1 CHR
190    GOSUB 230      : ' SET 2 BYTE VALUE
200   NEXT
210  POKE &H460,40-1 : ' SET WIDTH
220 END
230 '
240   POKE X,Y MOD 256:POKE X+1,Y ¥ 256
250  RETURN
```

ただし，CLS や [HOME CLR] をすると全画面が消去されますし，WIDTH 文を実行すると，表示が画面全体へ渡ってしまいます。

このプログラムの原理は，ワークエリアにある，行単位の属性をあらわすデータを書き直しているのです。

314,5	第0行のTOPアドレス	348,9	第13行のTOPアドレス
318,9	第1行のTOPアドレス	34C,D	第14行のTOPアドレス
31C,D	第2行のTOPアドレス	350,1	第15行のTOPアドレス
320,1	第3行のTOPアドレス	354,5	第16行のTOPアドレス
324,5	第4行のTOPアドレス	358,9	第17行のTOPアドレス
328,9	第5行のTOPアドレス	35C,D	第18行のTOPアドレス
32C,D	第6行のTOPアドレス	360,1	第19行のTOPアドレス
330,1	第7行のTOPアドレス	364,5	第20行のTOPアドレス
334,5	第8行のTOPアドレス	368,9	第21行のTOPアドレス
338,9	第9行のTOPアドレス	36C,D	第22行のTOPアドレス
33C,D	第10行のTOPアドレス	370,1	第23行のTOPアドレス
340,1	第11行のTOPアドレス	374,5	第24行のTOPアドレス
344,5	第12行のTOPアドレス		

各行のTOPアドレスが格納されています。

460H番地には，（1行の幅）-1の値が入ります。

ここでは使っていませんが，次の24コのアドレスには，それぞれ0行目と1行目，1行目と2行目，…，23行目と24行目がつながっているか，のフラグが入っています。

317，31B，31F
323，327，32B，32F
333，337，33B，33F
343，347，34B，34F
353，357，35B，35F
363，367，36B，36F
373，

フラグの意味は，

{ 80以上……行がつながっている。
  80未満……行がつながっていない。

です。

3-7 テキスト画面の2ページ目を利用

テキストVRAMの表示エリアは，A0000HからA0FFEHまでありますが，これと同じ形式の表示エリアがA1000HからA1FFEHまであります。また，アトリビュートエリアもA2000HからA2FFEHと同じ形式のものがA3000HからA3FFEHまであります。これらのエリアは，未使用となっています。

未使用では，もったいないので，ちょっとした有効利用を考えてみましょう。タイトル画面やメニュー表示など，表示しては消し，また表示する必要がある場合には，その画面をそっくりそのまま，未使用エリアにブロック転送して，保存しておくと良いでしょう。そして，必要なときに，本来のエリアに転送し直せばOKというわけです。

そのプログラム例を次に示します。A0000HからFFEHバイトの表示エリアをA1000Hからに転送し，次にA2000HからFFEHバイトのアトリビュートエリアをA3000Hに転送し，もとに戻すには，その逆を行うことになります。

8086にはストリング操作命令があり，一続きのメモリブロックの操作ができます。これにより，ブロック転送のほか，サーチ，比較といったオペレーションを実現することができます。ちなみに，次にブロック転送の公式を述べます。

ＤＳ＝転送元のセグメント

ＥＳ＝転送先のセグメント

ＳＩ＝転送元のオフセットアドレス（開始番地）

ＤＩ＝転送先のオフセットアドレス（開始番地）

ＤＦ＝0（ディレクションフラグを0としてインクリメントモードにする）

　　ＳＩ＝ＳＩ＋1：ＤＩ＝ＤＩ＋1となる。

ＣＸ＝転送するバイト数またはワード数

ＲＥＰ　くり返しプリフィックスでこれを使用すると次の転送命令がＣＸの値が0になるまで連続される。

ＭＯＶＳ　転送命令。MOVSBとするとバイト転送，MOVSWとするとワード（2バイト）転送となります。

次にBASICのCALL文で画面の退避・復帰を行うリストを示します。

画面の退避・復帰プログラム

```
1  ' save "VRAM.bas"
100 '=================================
110 ' DEF SEG=&H1F00
120 ' VRAM=&H0
130 ' A%=0:CALL VRAM(A%) ' Save
140 ' A%=1:CALL VRAM(A%) ' Restore
150 '=================================
```

```
160  DEF SEG=&H1F00
170 FOR AD=0 TO &H44
180  READ D$ : D=VAL("&H"+D$)
190  POKE AD,D
200 NEXT AD
210 END
220 DATA C4,37,26,8B,04,3D,00,00,74,06,3D,01,00,74,14,CF:'0000H
230 DATA B8,00,A0,BB,00,A1,E8,1D,00,B8,00,A2,BB,00,A3,E8:'0010H
240 DATA 14,00,CF,B8,00,A1,BB,00,A0,E8,0A,00,B8,00,A3,BB:'0020H
250 DATA 00,A2,E8,01,00,CF,8E,D8,8E,C3,33,F6,33,FF,FC,B9:'0030H
260 DATA FF,07,F3,A5,C3,00,00,00,00,00,00,00,00,00,00,00:'0040H
```

画面を退避させるときは，A%＝0でコールし，復活させるときはA%＝1としてコールします。

3-8 ひらがなの表示

セグメント0の2B4H～2B7Hに格納されているアドレス(オフセット，セグメント)をaとすると，a－420Hからa－1には，ひらがなフォントデータが格納されています。

次に示すプログラムを走らせると，そのイメージを表示してくれます。

ひらがなフォント表示プログラム

```
0 'SAVE"HIRAGA"
100 '
110 '  HIRAGANA OUTPUT FROM ROM
120 '
130  OPEN "SCRN:" AS 1
140  DEF SEG=0
150  DIM D(15)
160   A=PEEK(&H2B4)+PEEK(&H2B5)*256
170   SG=PEEK(&H2B6)+PEEK(&H2B7)*256
180  DEF SEG=SG
190   DT=A-&H420
200  FOR I=0 TO 15
210   B=PEEK(DT+I)
220    IF B - 128>=0THEN B=B-128:PRINT #1,"**"; ELSE PRINT #1,"  ";
230    IF B - 64>=0 THEN B=B-64 :PRINT #1,"**"; ELSE PRINT #1,"  ";
240    IF B - 32>=0 THEN B=B-32 :PRINT #1,"**"; ELSE PRINT #1,"  ";
250    IF B - 16>=0 THEN B=B-16 :PRINT #1,"**"; ELSE PRINT #1,"  ";
260    IF B - 8 >=0 THEN B=B-8  :PRINT #1,"**"; ELSE PRINT #1,"  ";
270    IF B - 4 >=0 THEN B=B-4  :PRINT #1,"**"; ELSE PRINT #1,"  ";
280    IF B - 2 >=0 THEN B=B-2  :PRINT #1,"**"; ELSE PRINT #1,"  ";
290    IF B - 1 >=0 THEN        :PRINT #1,"**"; ELSE PRINT #1,"  ";
300   PRINT #1,""
310  NEXT I
320  DT=DT+15
330   IF DT-15<A THEN 200
340 END
```

《実行結果の一部》　次に表示結果の一部を示します。

これらのひらがなは，したがって，漢字ROMがなくても表示できます。ただし，テキスト画面にはできませんが，グラフィック画面には表示することができます。次に，それらを表示するプログラムを示します。

ひらがな表示プログラム

```
0 'SAVE "HIRA.DSP"
100 '
110 '  DISPLAY HIRAGANA
120 '   WITHOUT KANJI ROM
130 '
140  ROLL 199:CLS
150  FOR I=0 TO &H9F-&H7C
160   PUT (10+I*12,10),KANJI(&H7C+I),PSET,7,0
170  NEXT
180  FOR I=0 TO &HFF-&HE0
190   PUT (10+I*12,30),KANJI(&HE0+I),PSET,7,0
200  NEXT
```

文字とコードの対応表は，PC-9801　USER'S　MANUAL をご覧下さい。

ただし，コード7CH（¦）と7FH（~）は，テキスト画面にも表示できます。

```
? CHR$(&H7C)CHR$(&H7E)
¦~
```

3-9　TABキーとTAB関数

TABキーのTABの意味とTAB関数のTABの意味が異なります。

まず，TAB関数の意味を調べるために，次に示すようなことをして下さい。

```
? "A"TAB(2)"B"TAB(10)"C"TAB(13)"D"
A B       C  D
Ok
123456789+123
```

結果をみると分かるように，行の最初の文字を0カラムとして，TABの示すカラムまでスキップせよ，ということです。

では，TABキーの方はどうかと申しますと，次のようにするとその正体が分かります。

行の一番左はしにカーソルをもってきて，TABキーを押して下さい。一定の位置で止まりますね。

```
0123456789+123456789+123456789+123456789+123456789+123456789+123456789+123456789
■       ■       ■       ■       ■       ■       ■       ■       ■       ■       ■
```

このように，TABキーは，8カラム目ごとにスキップするのです。

カーソルがnカラム目にあるとき，TABキーをおすと，カーソルは，

n＋8－(n mod 8)カラム目

に進みます。ただし，行の右端で止まります。

しかし，このようにTABを固定すると不便な場合があります。例えば，次に示すように，アセンブル結果を表示する場合，8カラム目ごとのTABでは使いものになりません。

```
0123456789+123456789+123456789+123456789+
0000 FC             CLD
0001 16             PUSH    SS
0002 1F             POP     DS
0003 A1A406         MOV     AX,[06A4]
0006 050400         ADD     AX,0004
0009 A3EA06         MOV     [06EA],AX
000C BF4700         MOV     DI,0047
000F CDC4           INT     C4
0011 CF             IRET
```

この例では，

○アドレスを　　　　0カラム目から

○オブジェクトを　　5カラム目から

○ソース・リストを　20カラム目から

{ オペコードを　20カラム目から
{ オペランドを　28カラム目から

というようにスキップするカラム数がまちまちです。このカラム数のスキップは表示を見やすくするものですから自分で設定できた方が便利です。

このようなわけでTAB関数は，その引数で，このTAB位置を設定するようになっているのです。

先程のアセンブルリストの表示を実現するサンプルプログラムを次に示しました。

TAB設定プログラム

```
0 'SAVE"TAB4.TST"
100 '
110 '   TAB SAMPLE
120 '
130  ADRS$="0003"
140  OBJ$ ="A1A406"
150  OPCODE$="MOV"
160  OPRAND$="AX,[06A4]"
170 '
180   GOSUB *DSP.LINE
190 '
200 END
210 '
220 *DSP.LINE
230  PRINT ADRS$;
```

```
240  PRINT TAB(5)OBJ$;
250  PRINT TAB(20)OPCODE$;
260  PRINT TAB(28)OPRAND$
270 RETURN
```

220行から270行がサブルーチンになっていて，

ADRS$………アドレス

OBJ$…………オブジェクト

OPCODE$…オペコード

OPRAND$…オペランド

をセットして，＊DSP.LINEをGOSUBすると，TABを設定して表示するようになっています。

次が実行結果です。

```
0003 A1A406          MOV     AX,[06A4]
```

第４章　グラフィック画面

4-1　G-VRAM
　4-1-1　G-VRAMのメモリマップ
　4-1-2　高速画面クリア
4-2　カラーパレット
　4-2-1　機械語によるカラーパレットの制御
　4-2-2　カラーパレットの初期化
4-3　ボーダーカラー
4-4　グラフィックBIOSとGDC
　4-4-1　グラフィックBIOSのワークエリア
　4-4-2　PSET　ドットを打つ
　4-4-3　ドットを読み出す
　4-4-4　直線・箱型を描く　LINE
　4-4-5　円弧を描く　CIRCLE
　4-4-6　グラフィックパターンを描く
　4-4-7　高速書き込みモードにする
4-5　マシン語によるG-VRAM直接アクセス法
4-6　グラフィックLIOとBASICコマンド
4-7　3Dパッケージの紹介

第4章　グラフィック画面

4-1　G-VRAM

4-1-1 G-VRAMのメモリマップ

グラフィックVRAMは，主記憶空間の物理アドレス，A8000H番地からBFFFFH番地までの32Kバイト×3プレーンに割り当てられています。

G-VRAMは，PC-8801のようなバンク切り換えではなく，主記憶空間内にアドレッシングされていますので，マシン語はもとより直接BASICからもアクセスできます。

このG-VRAMは，画面モードによって，さらに何枚かのプレーンに分割されます（図4-1-1)。

テキスト画面は，GDC μPD7220によって制御されていると述べましたが，グラフィック画面もGDC μPD7220によって制御されています。グラフィック画面とテキスト画面にそれぞれ1個ずつ合計2個のGDCを使っているのです。

GDCには，特別にワード（2バイト1組の）アドレッシングを行いますが，それも同時に書

<table>
<tr><th rowspan="2">GDC
アドレス
(16進)</th><th rowspan="2">CPU
アドレス
(16進)</th><th colspan="2">640×200ドット</th><th colspan="2">640×400ドット</th></tr>
<tr><th>モノクロ</th><th>カラー</th><th>モノクロ</th><th>カラー</th></tr>
<tr><td>4000
↕8000ワード
5F3F</td><td>A8000
↕16000バイト
ABE7F</td><td>プレーン
1
G-VRAM0</td><td>プレーン1
青</td><td rowspan="2">プレーン
1</td><td rowspan="2">プレーン1
青</td></tr>
<tr><td>5F40
↕8000ワード
7E7F</td><td>ABE80
↕16000バイト
AFCFF</td><td>プレーン
4
G-VRAM1</td><td>プレーン2
青</td></tr>
<tr><td>7E80
↕380ワード
7FFF</td><td>AFD00
↕760バイト
AFFFF</td><td>未使用</td><td>未使用</td><td>未使用</td><td>未使用</td></tr>
<tr><td>8000

9F3F</td><td>B0000

B3E7F</td><td>プレーン
2
G-VRAM2</td><td>プレーン1
赤</td><td rowspan="2">プレーン
2</td><td rowspan="2">プレーン1
赤</td></tr>
<tr><td>9F40

BE7F</td><td>B3E80

B7CFF</td><td>プレーン
5
G-VRAM3</td><td>プレーン2
赤</td></tr>
<tr><td>BE80

BFFF</td><td>B7D00
↕760バイト
B7FFF</td><td>未使用</td><td>未使用</td><td>未使用</td><td>未使用</td></tr>
<tr><td>C000

DF3F</td><td>B8000

BBE7F</td><td>プレーン
3
G-VRAM4</td><td>プレーン1
緑</td><td rowspan="2">プレーン
3</td><td rowspan="2">プレーン1
緑</td></tr>
<tr><td>DF40

FE7F</td><td>BBE80

BFCFF</td><td>プレーン
6
G-VRAM5</td><td>プレーン2
緑</td></tr>
<tr><td>FE80

FFFF</td><td>BFD00
↕760バイト
BFFFF</td><td>未使用</td><td>未使用</td><td>未使用</td><td>未使用</td></tr>
</table>

図4-1-1 G-VRAMのアドレスと働き

いておきました。

1ビットで1ドットを表示するので，640×200ドットの場合，

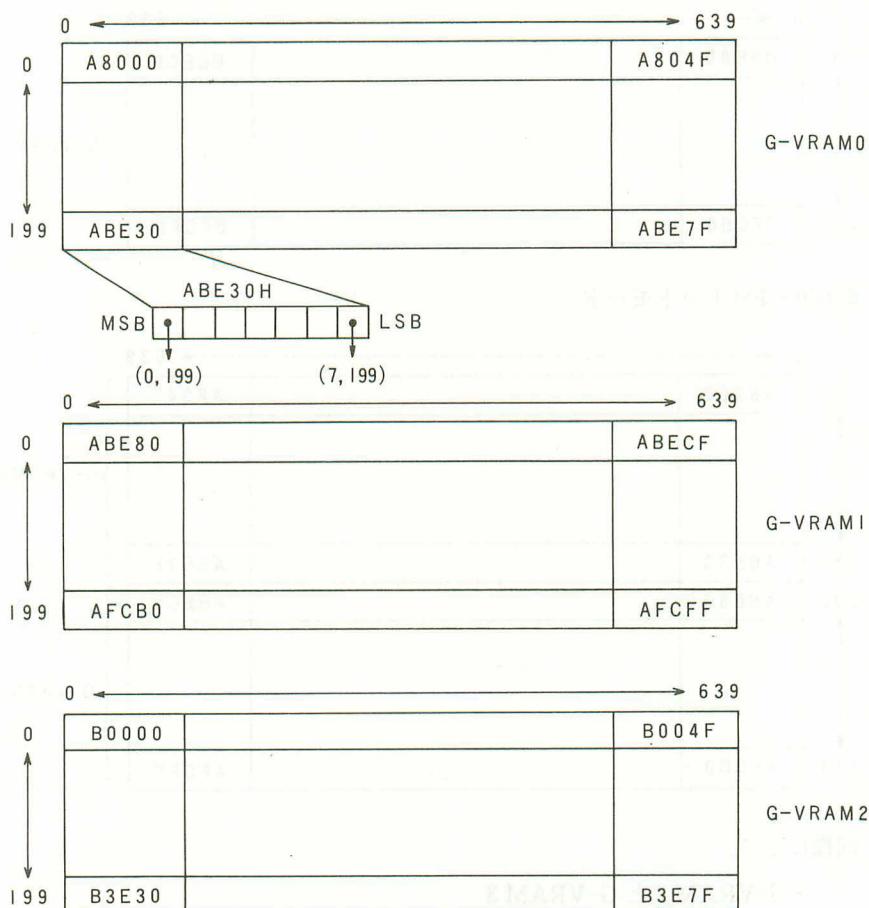
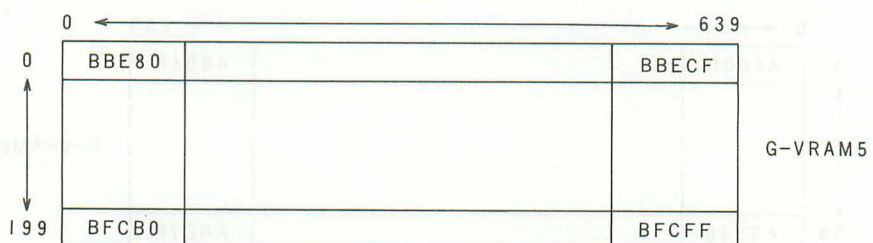
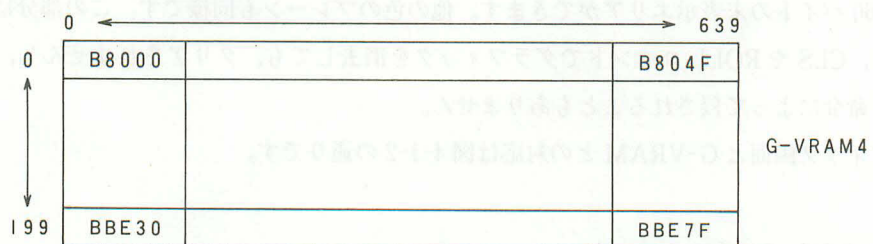
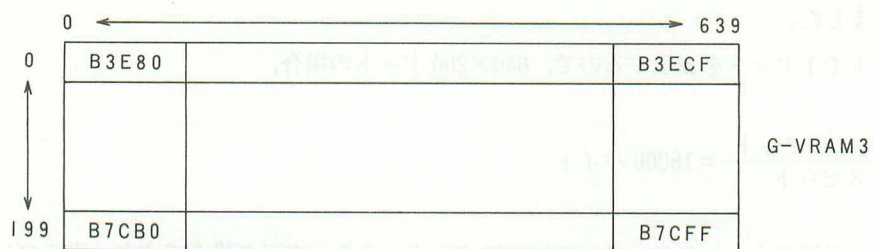
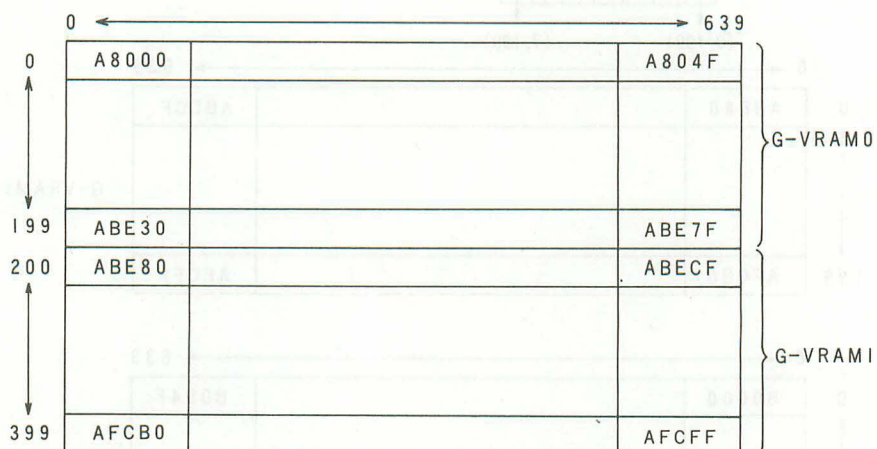
$$\frac{640\times200\text{ドット}}{8\text{ビット}}=16000\text{バイト}$$

のメモリが必要です。したがって，640×200ドット，カラー表示の場合ですと，青のプレーンのうしろに760バイトの未表示エリアができます。他の色のプレーンも同様です。この部分は，未使用エリアで，CLSやROLLコマンドでグラフィックを消去しても，クリアされませんし，他のグラフィック命令によって侵されることもありません。

グラフィック画面とG-VRAMとの対応は図4-1-2の通りです。

	0		639	
0	B3E80		B3ECF	G-VRAM3
199	B7CB0		B7CFF	

	0		639	
0	B8000		B804F	G-VRAM4
199	BBE30		BBE7F	

	0		639	
0	BBE80		BBECF	G-VRAM5
199	BFCB0		BFCFF	

※ 640×400 ドットモード

同様にして,

・G-VRAM 2 と G-VRAM 3

・G-VRAM 4 と G-VRAM 5

が結集して縦 400 ドットを形成します。

図 4-1-2 グラフィック画面と G-VRAM との対応

4-1-2 高速画面クリア

グラフィック画面を消去する場合，通常

　　ＣＬＳ　２

を用いますが，ROLL コマンドを用いて，

　　ＲＯＬＬ　１９９：ＲＯＬＬ　１　　←640×200 ドットの場合

とすると画面全体を消去できます。CLS より，約 30 倍速く画面をクリアすることができます。ただし，ROLL コマンドを用いますと，VIEW ポートの値は無視され，すべての G-VRAM がクリアされますので，その点使い分けて下さい。

ROLL について説明しますと，

　　ＲＯＬＬ　ｎ

は，グラフィック画面全体を n ドットスクロールアップします。したがって，画面のたてドット数だけスクロールアップさせますと，画面範囲外に出ていって，最終的にクリアされたのと同じになるわけなのです。

ただし，たて 200 ドットの場合，

　　　１≦　ｎ　≦　１９９

なので，先程のように，200 ドットすべてをスクロール･アップするには，ROLL 199 と ROLL 1 の 2 回に分けて行う必要があったのです。ただし，PC-9801F では，CLS 2 で消しても高速に画面が消去できます。

4-2　カラーパレット

4-2-1 マシン語によるカラーパレットの制御

BASIC ならば，

　　ＣＯＬＯＲ＝（〈パレット番号〉，〈カラーコード〉）

とすることによって，カラーパレットをセットすることができます。

例えば，

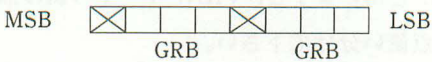
　　ＣＯＬＯＲ＝（４，５）

とすれば，カラーパレット 4 にカラーコード 5 が設定されます。こうすると，今まで，LINE や CIRCLE 等でパレット番号 4 に設定して描いたものが，すべて，水色になります。ただし，これは，テキスト画面の色には効果がありません。

これをマシン語で制御するには，OUT 命令で行います。

カラーパレットには，I/O ポートＡ８Ｈ～AEH が割当てられ，次の表のように対応しています。

OUTPUTアドレス	パレット番号	
	上位4ビット	下位4ビット
A8H	3	7
AAH	1	5
ACH	2	6
AEH	0	4

MSB ☒ GRB ☒ GRB LSB

カラーパレットのI/Oポート

パレット番号に対するカラーコードの指定は，上の各ポートのGRBのビットに，次のデータを出力すればよいわけです。

データ	カラーコード	
GRB	コード	色
000	0	黒
001	1	青
010	2	赤
011	3	紫
100	4	緑
101	5	水色
110	6	黄
111	7	白

カラーコードの指定

例えば，

COLOR=（2，4）

は，

OUT &HAC,&H4X

（ただし，Xはパレット番号6に対応するカラーコード）

とすればよいわけです。ただし，1つのI/Oポートで2つのカラーパレットに対応していますので，一方のカラーパレットだけを変えたい場合は，前に出力した値を覚えておかなければなりません。

BASICでは，ワークエリアにその情報が格納されており，次の表のようになっています。

セグメント60H	パレット番号	
	上位 4ビット	下位 4ビット
644H	6	7
645H	4	5
646H	2	3
647H	0	1

カラーパレットの情報

したがって先程の例は,

```
DEF SEG=&H60
OUT &HAC,&H40+(PEEK(&H644) AND &HF)
POKE &H646,PEEK(&H646) AND &HF0 OR 4
```

とすればよいでしょう。

マシン語では次のようになります。

```
0000 A04406          MOV       AL,[0644]; パレット番号67をとってくる
0003 240F            AND       AL,0F    ; パレット番号6の分だけ残す
0005 0C40            OR        AL,40    ; パレット番号2にセットするべきカラーコード
0007 E6AC            OUT       AC,AL
0009 A04606          MOV       AL,[0646]; ワークエリアのパレット番号2に対応する
000C 24F0            AND       AL,F0    ; ところを更新する
000E 0C04            OR        AL,04
0010 A24606          MOV       [0646],AL
```

BASICのROM内ルーチンを利用する場合は，次のようにします。

```
0000 16              PUSH      SS
0001 1F              POP       DS
0002 A04606          MOV       AL,[0646] ┐
0005 240F            AND       AL,0F     │ パレット番号セット
0007 0C40            OR        AL,40     │ COLOR=(2,4)に相当
0009 A24606          MOV       [0646],AL ┘
000C BB4006          MOV       BX,0640   ; ROM内ルーチンコール用前処理
000F B443            MOV       AH,43
0011 CD18            INT       18
0013 CF              IRET
```

パレットの色が本当に変化するかのチェックに，次のプログラムを走らせて，8色8本の直線をひいておくとよいでしょう。

```
10  CONSOLE ,,,1
20  FOR I=0 TO 7
30   LINE (320,0)-(639,100+I*5),I
40  NEXT
50 END
```

4-2-2 カラーパレットの初期化

カラーパレットは使って便利なものではあるのですが，1つのプログラムでカラーパレットを操作した後，別のプログラムで，思ったとおりの色がでなかったりするということがあります。これは，カラーパレットを初期化すれば直ります。カラーパレットのカラーコードを変更した場合，そのプログラムの終わりに，カラーパレットを初期化するようにしましょう。

カラーパレットの初期化は，ダイレクトモードかプログラム中で，

```
FOR I=0 TO 7:COLOR=(I,I):NEXT
```

とすればよいです。

これをマシン語でやれば，次のようになります。

```
0000 16            PUSH    SS
0001 1F            POP     DS
0002 BB4406        MOV     BX,0644
0005 B86745        MOV     AX,4567
0008 8907          MOV     [BX],AX
000A B82301        MOV     AX,0123
000D 43            INC     BX
000E 43            INC     BX
000F 8907          MOV     [BX],AX
0011 BB4006        MOV     BX,0640
0014 B443          MOV     AH,43
0016 CD18          INT     18
0018 CF            IRET
```

4-3 ボーダーカラー

ボーダーカラーは，画面のまわりの色のことで，COLOR 文の3番目のパラメーターで制御します。

ボーダーカラーをマシン語で制御するには，I/Oポート6CHのb_4～b_6の3ビットを使います。

○ OUTPUT ポート　6CH

○データ

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
MSB	0	G	R	B	0	0	0	0	LSB

カラーコードを出力

黒	0 0 H
青	1 0 H
赤	2 0 H
紫	3 0 H
緑	4 0 H
水色	5 0 H
黄	6 0 H
白	7 0 H

ボーダーカラーのI/Oポート

COLOR文の第3パラメータで指定した場合，このOUTPUTデータが，セグメント60Hのアドレス641Hに入れられます。

4-4 グラフィックBIOSとGDC(Graphic Display Controller)

PC-9801には，グラフィック描画にGDC μPD 7220というLSIを用いています。

このLSIは，テキスト画面に高速多機能に文字を描くのにも使うことができ，PC-9801には，グラフィック用とテキスト用の2個のGDCが使われております。

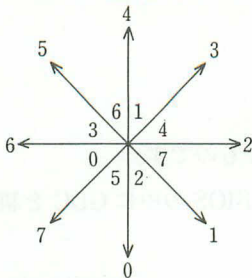
GDCを用いると，高解像度画面に，高速に直線，円弧，箱型等を描画することができますが，そのデータのセット，タイミング等の処理が面倒です。

例えば，直線ですと，始点・終点を指定するのではなく，

○始点

○方向

○長さ

を指定しなければならず，この計算がなかなか面倒なものです。

しかし，心配はいりません。N_{88}-BASIC（86）のBIOSの中にGDCを制御するプログラムが用意されています。

ここでは，GDCを直接制御するのではなく，このBIOSの使い方を説明しましょう。

4-4-1 グラフィックBIOSのワークエリア

グラフィックBIOSのワークエリアは，セグメント60Hの640Hからです。表4-4-1にして，その意味をまとめています。

このワークエリアに値をセットして，AHにBIOSコマンド・コードをセットして，INT 18Hによって，グラフィックBIOSを使用します。

640H	GBON-PTN　カラーパレットを設定する。 下位3ビットに色の情報を書き込む 0　黒　(デフォルト) 1　青　(デフォルト) 2　赤　(デフォルト) 3　紫　(デフォルト) 4　緑　(デフォルト) 5　水色　(デフォルト) 6　黄　(デフォルト) 7　白　(デフォルト)
641H	GBBCC　ボーダーカラーを設定する。 00　黒　(デフォルト) 10　青 20　赤 30　紫 40　緑 50　水色 60　黄 70　白
642H	GBDOTU　1つのプレーンだけを処理するときのモード。 0　リプレース　(置き換え) 1　コンプリメント　(XOR) 2　クリア　(NOT) 3　セット　(OR) 2のクリアは描画パターンの1に対応する画面上のドットを0にします。 3のセットはクリアと逆に,描画パターンの1に対応する画面上のドットを1にします。
643H	GBDSP　描画方向の指定 4　5　3　6 1　3　4　6　2　0　7　5 2　7　1　0 0～7の数字で，描く方向を指定します。方向と数字との対応は，左図のとおりです。 矢印先端の数字は箱型 矢印中心の数字は円孤
644H ≀ 647H	カラーパレットの章を参照して下さい。

648, 9H	GBSX1　描画始点のX座標
64A, BH	GBSY1　描画始点のY座標
64C, DH	GBLNG1　何ドット描くかのドット数の指定
64E, FH	GBWDPA 描画パターン・バッファのスタートオフセットアドレス (セグメントアドレスはESレジスタで指定します。)
650, 1H	GBRBUF1 描画画面からドット情報をよみ出す場合のバッファ1（プレーン1,4用の）の 先頭オフセットアドレス。
652, 3H	GBRBUF2　上と同じ。ただし，プレーン2,5用。
654, 5H	GBRBUF3　上と同じ。ただし，プレーン3,6用。
656, 7H	GBSX2　LINEの終点のX座標
658, 9H	GBSY2　LINEの終点のY座標
65A, BH	GBMDOT　マスキングドット数 円弧の描画時に使用する。 円弧描画の章を参照。
65C, DH	GBCIR　円の半径
65E, FH	GBLNG2 グラフィック文字の書き込みの場合の（たて方向のドット数）－1 ただし，8×8ドット文字の場合は，0を入れる。
660, 1H	GBLPTN　線種パターン 直線，円弧描画のときの線の情報 BASICのラインスタイルと同等。
662H ≀ 667H	GBDOTI 8×8ドット　グラフィック文字基本パターンバッファ。
668H	GBDTYP　描画タイプ 1　直　線 2　矩　形　(BOX) 4　円　孤
1110H ≀	グラフィックパターンのバッファとして使う。 GBRBUF1～3のバッファとして使用するとよい。

表4-4-1 グラフィックBIOSのワークエリア

4-4-2 PSET ドットをうつ

AHにBIOSコード45Hをセットし，CHに，以下の情報（図4-4-2）をセットします。

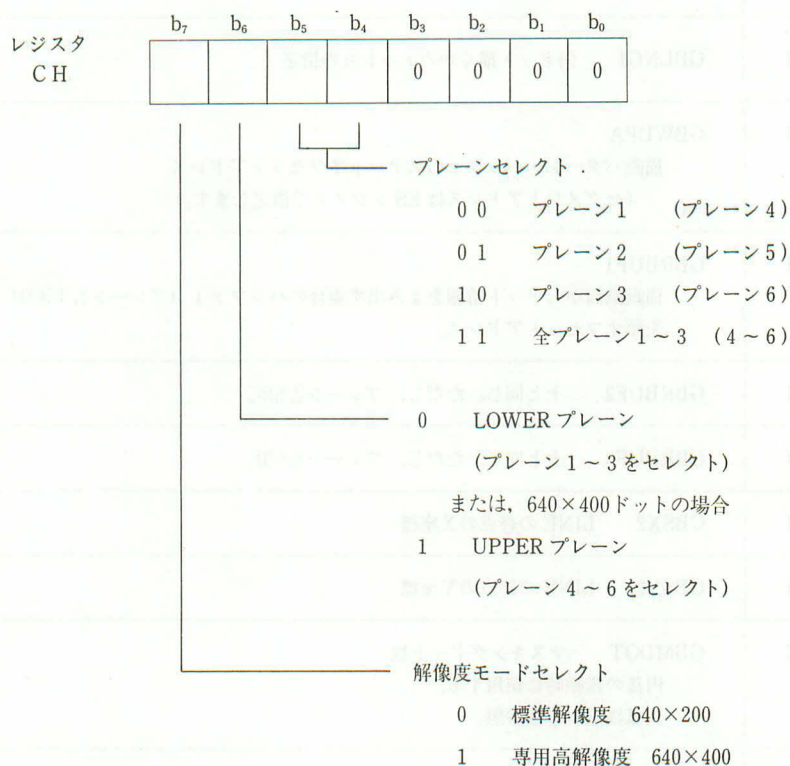



図4-4-2 ドット情報セット

例えば，640×200ドットモードで，プレーン1～3に同時に書き込む場合は，

$b_7 b_6 b_5 b_4 b_3 b_2 2_1 b_0$

0 0 1 1 0 0 0 0 B

16進表記すれば，30HをCHレジスタにセットすればよいことになります。

ESレジスタに描画パターンバッファのセグメントアドレスをセットします。N_{88}-BASIC(86)のワークエリアを利用するなら，セグメント60Hのオフセット1110Hを指定して下さい。

DSレジスタに60Hをセットする。

BXレジスタに640Hをセットする。

表4-1-1で示したBIOSのワークエリアの中で表4-4-2のものを指定する。

GBON-PTN	640H	カラーパレット番号
GBDOTU	642H	CH レジスタの b_4 b_5に 00, 01, 10 を指定したときに ドットをうつモードを指定
GBSX 1	648, 9H	X座標
GBSY 1	64A, BH	Y座標
GBLNG 1	64C, DH	ドット数。1点だけうつならば 1をセットする。
GBWDPA	64E, FH	描画パターンバッファの開始 オフセットアドレス 1110H を指定すればよい。

表 4-4-2 BASICのワークエリアを利用

PSET (X, Y), Cとして使うなら,

(648, 9H)=X

(64A, BH)=Y

(640H)=C

(64C, DH)=1

(64E, FH)=1110H

(1110, 1H)=FFH

とすればよいわけです。

このコマンドは, 1点をうつ以外にも, GBLNG1 (64C, DH) にうつドット数を指定することにって, 連続的にドットをうつこともできます。ただし, 指定できるドット数は,

図 4-4-3 ドットの指定数

図 4-4-3 に示すように, 始点 (X, Y) から最終点までのドット数までです。それ以上を指定すると暴走することがあります。しかし, 連続してドットをうつ方が, 1ドット当たりの描画時間は極端に短くなります。例えば, 1ドットだけの場合 2 msec 程ですが, 1000 ドット連続に描いた場合 1

ドット当たり 0.02 msec にもなります。

PSET (200, 100), 7 と同等の処理をマシン語で行った例を次に示します。

PSET (200, 100), 7 と同等の処理プログラム

```
0000 16          PUSH   SS  ┐
0001 16          PUSH   SS  │
0002 1F          POP    DS  ├ DS=ES=SS=60H にする 注)
0003 07          POP    ES  ┘
0004 B007        MOV    AL,07; カラーパレット=7 通常白（ここを0にするとPRESETになる）
0006 A24006      MOV    [0640],AL
0009 B8C800      MOV    AX,00C8 ; X=200
000C A34806      MOV    [0648],AX
000F B86400      MOV    AX,0064 ; Y=100
0012 A34A06      MOV    [064A],AX
0015 B80100      MOV    AX,0001 ; 1ドット
0018 A34C06      MOV    [064C],AX
001B B81011      MOV    AX,1110 ; 描画パターンのバッファ先頭オフセットアドレス
001E A34E06      MOV    [064E],AX
0021 B8FF00      MOV    AX,00FF ; 描画パターン
0024 A31011      MOV    [1110],AX
0027 B530        MOV    CH,30   ; 640×200モード，プレーン1～3
0029 BB4006      MOV    BX,0640 ┐
002C B445        MOV    AH,45   ├ BIOS コマンド実行
002E CD18        INT    18      ┘
0030 31C0        XOR    AX,AX   ;
0032 CF          IRET
```

カラーパレット 640 H は，この例では，7 ですが，00 H にすると，PRESET となります。

INT 18 H 前のすべてのレジスタは壊されません。

注)モニタに入ったときとか，USR，CALL でマシン語プログラムを読んだ際には，SS レジスタにはテキストセグメントの値 60 H が入っています。

4-4-3 ドットを読み出す

4-4-2 の PSET は画面にドットをうち出しましたが，今度は，その逆に，画面上のドットパターンをワークエリア上のバッファに読み込む命令です。

AH レジスタに BIOS コード 46 H をセットし，CH レジスタ DS, BX, ES レジスタは，4-4-2PSET と同様の内容をセットし，BIOS のワークエリアの中で表 4-4-3 のものを指定します。

GBSX1	648，9H	読み込み始点のX座標
GBSY1	64A，BH	読み込み始点のY座標
GBLNG1	64C，DH	始点から何点読み込むかのドット数。
GBRBUF1	650，1H	読み込んだドット情報を格納するバッファ1のオフセットアドレス。 CHレジスタのb_6が， 0…プレーン1 1…プレーン4 から読み込む。
GBRBUF2	652，3	読み込みバッファ2。 CHレジスタのb_6が， 0…プレーン2 1…プレーン5 から読み込む。
GBRBUF3	654，5	読み込みバッファ3。 CHレジスタのb_6が， 0…プレーン3 1…プレーン6 から読み込む。

表4-4-3A ドット・リードBIOSワークエリア

次にサンプルプログラムを示します。

16ドット連続リードプログラム

```
0000 16          PUSH   SS   }
0001 16          PUSH   SS   }
0002 1F          POP    DS   }  DS=ES=SS=60H
0003 07          POP    ES   }
0004 B8C800      MOV    AX,00C8 ; X=200
0007 A34806      MOV    [0648],AX
000A B86400      MOV    AX,0064 ; Y=100
000D A34A06      MOV    [064A],AX
0010 B81000      MOV    AX,0010 ; 16ビット読み込む。
0013 A34C06      MOV    [064C],AX
0016 B81011      MOV    AX,1110 ; 読み込みバッファ1のオフセットアドレス
0019 A35006      MOV    [0650],AX
001C 051000      ADD    AX,0010 ; 読み込みバッファ2のオフセットアドレス
001F A35206      MOV    [0652],AX
0022 051000      ADD    AX,0010 ; 読み込みバッファ3のオフセットアドレス
0025 A35406      MOV    [0654],AX
0028 B530        MOV    CH,30   ; 640×200モード。プレーン，1～3。
002A BB4006      MOV    BX,0640 ; BIOSのワークエリアのオフセットアドレス。
002D B446        MOV    AH,46   }  BIOS CALL
002F CD18        INT    18      }
0031 31C0        XOR    AX,AX   ; USR等でよんだときエラーがでないため。
0033 CF          IRET
```

座標(200, 100)から連続して16ドットプレーン1～3から同時に読み込むプログラムです。バッファは，セグメント60Hの

1110H………GBRBUF1

１１２０H………GBRBUF２

　１１３０H………GBRBUF３

としました。

まずモニタで，

　h］C６０

　h］F１１１０，１１４０，０

と，バッファを0クリアして，BASICにもどり，次のBASICプログラムを走らせてから，先程のサンプルプログラムを走らせて下さい。

```
10 LINE (200,100)-(220,100),7,,&HA5A5
```

モニタでバッファの中身を確かめてみましょう。

```
h]C60
h]D1110,1140
1110 A5 A5 00 00 00 00 00 00 00 00 00 00 00 00 00 00          ..
1120 A5 A5 00 00 00 00 00 00 00 00 00 00 00 00 00 00          ..
1130 A5 A5 00 00 00 00 00 00 00 00 00 00 00 00 00 00          ..
1140 00
```

確かに，LINEのパターンＡ５Ａ５Ｈが入っていますね。上から順に，プレーン１，２，３の情報です。今は，カラーパレット番号に7を指定したので，３つともＡ５Ａ５Ｈになったのです。

では，BASICに戻って，一旦，画面を

```
ROLL 199
```

で消去したあと，今度は，次のBASICプログラムを走らせてみて下さい。今度は，カラーパレットを5にしました。

```
10 LINE (200,100)-(220,100),5,,&HA5A5
```

ここで，先程のサンプルプログラムを走らせて下さい。

今度はバッファにどのような値が入ったでしょうか？ さっきと同じようにして，モニタで見ますと，

```
h]C60
h]D1110,1140
1110 A5 A5 00 00 00 00 00 00 00 00 00 00 00 00 00 00          ..
1120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1130 A5 A5 00 00 00 00 00 00 00 00 00 00 00 00 00 00          ..
1140 00
```

となっていました。プレーン２に対応する部分が，0000Ｈですね。カラーパレット５というのは，通常，水色ですが，これは，プレーン１（通常，青）とプレーン３（通常，緑）を合成して作るからなのです。つまり，このとき，プレーン２は使用しないからなのです。ここで，色とプレーンの

関係について説明しておきましょう。

プレーン１（４）……青　　　００１Ｂ＝１

プレーン２（５）……赤　　　０１０Ｂ＝２

プレーン３（６）……緑　　　１００Ｂ＝４

が光の三原色に対応していて，この三原色合成で，他の５色も表示するのです。

プレーン１～３でページ１

プレーン４～６でページ２

を形成します。カラーパレットの章で示したGRB（Green・Red・Blue）のビットとカラーコード，色の関係は，プレーンで表わせば，次表のようになります。ただし，これは，640×200ドットカラーモードで，640×400ドットカラーモードのときは，ページ１と２を組み合わせて，１ページとします。対応する色は同じです。白黒モードの場合は，色の合成としての意味はなく，プレーンの合成としての意味となります。

ＧＲＢ	カラーコード	色	色の合成	プレーン
０００	０	黒	発光させない。	全プレーン表示しない。
００１	１	青	三原色の１つ。	プレーン１（４）
０１０	２	赤	三原色の１つ。	プレーン２（５）
０１１	３	紫	青（１）と赤（２）の合成で，１＋２＝３	プレーン１（４）とプレーン２（５）の合成
１００	４	緑	三原色の１つ。	プレーン３（６）
１０１	５	水　色	青（１）と緑（４）の合成で，１＋４＝５	プレーン１（４）とプレーン３（６）の合成
１１０	６	黄	赤（２）と緑（４）の合成で，２＋４＝６	プレーン２（５）とプレーン３（６）の合成
１１１	７	白	青（１）と赤（２）と緑（４）の合成で，１＋２＋４＝７	全プレーンの合成１と２と３（４と５と６）

表4-4-3 B GRBのビットとカラーコードとプレーン　ただし，（　）の中はページ２のとき。

4-4-4 直線，箱型を描くLINE

BASICのLINEステートメントに相当する命令です。

AHレジスタにBIOSコード47Hをセットします。

CH，DS，BXレジスタは，4-4-2と同様の内容をセットします。

BIOSのワークエリアの中で表4-4-4のものを指定します。

GBON-PTN	６４０Ｈ	カラーパレット番号
GBDOTU	６４２Ｈ	CH レジの b_5 b_4＝００，０１，１０を指定したとき，つまり，単一画面にだけ直線をひくときの描画オペレーションモード。
GBSX 1	６４８，９Ｈ	始点のＸ座標
GBSY 1	６４Ａ，ＢＨ	始点のＹ座標
GBSX 2	６５６，７Ｈ	終点のＸ座標
GBSY 2	６５８，９Ｈ	終点のＹ座標
GBLPTN	６６０，１Ｈ	ラインスタイル
GBDTYP	６６８Ｈ	1……直線 2……箱型（LINEのＢ指定に相当）

表 4-4-4 LINE の BIOS ワークエリア

次にサンプルプログラムとして，

LINE (300, 10)－(620, 100), 7,, &HFFFF を実行したプログラムを示します。

LINE (300,10)－(620,100),7,,&HFFFF

```
0000 16          PUSH   SS }
0001 1F          POP    DS }  DS=SS=60H
0002 B007        MOV    AL,07    ; カラーパレット＝7
0004 A24006      MOV    [0640],AL
0007 B82C01      MOV    AX,012C  ; X1=300
000A A34806      MOV    [0648],AX
000D B80A00      MOV    AX,000A  ; Y1=10
0010 A34A06      MOV    [064A],AX
0013 B86C02      MOV    AX,026C  ; X2=620
0016 A35606      MOV    [0656],AX
0019 B86400      MOV    AX,0064  ; Y2=100
001C A35806      MOV    [0658],AX
001F B8A5A5      MOV    AX,A5A5  ; ラインスタイル
0022 A36006      MOV    [0660],AX
0025 B001        MOV    AL,01    ; 直線
0027 A26806      MOV    [0668],AL
002A B530        MOV    CH,30    ; 640×200
002C BB4006      MOV    BX,0640
002F B447        MOV    AH,47 }
0031 CD18        INT    18    }  BIOS CALL
0033 31C0        XOR    AX,AX
0035 CF          IRET
```

ただし，箱型を描く場合は，

GBDSP（643 H）

描画方向の指定をしなければなりません。方向と数字については，表 4-4-1 グラフィック BIOS のワークエリアを参照して下さい。

方向に対して，どのような箱型が描かれるかを例をあげて説明しますと，図 4-4-4 A のような位置に 2 点を指定し，（始）と書いてある点を始点としますと，直線ならば，1 本に決まります。

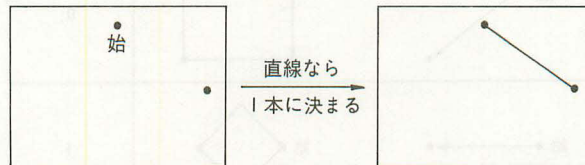



図 4-4-4 A　2 点と直線

しかし，箱型の場合，GBDSP（描き出す方向）を適切に決めないと，図 4-4-4 B のような箱型を描いてしまいます。

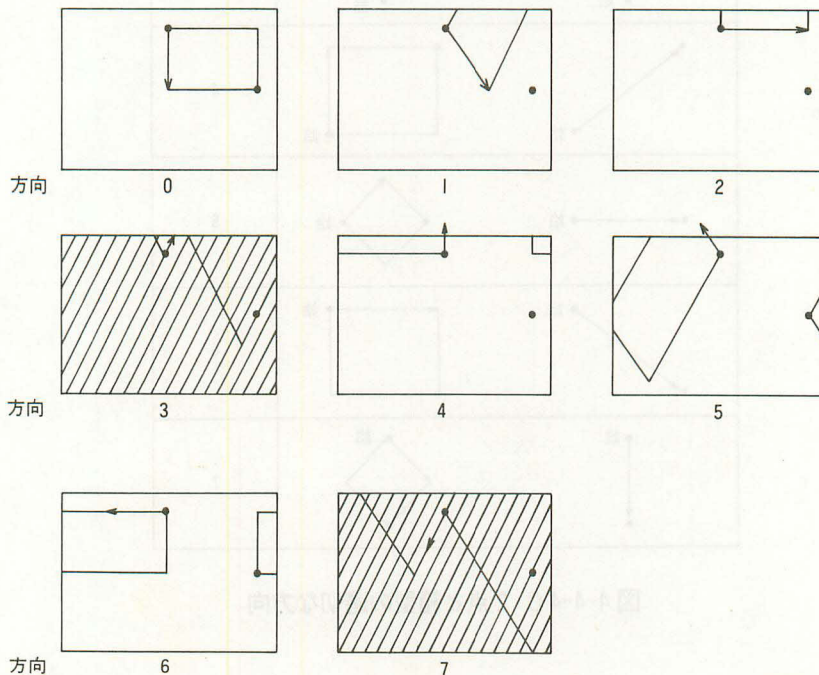



図 4-4-4 B　2 点と箱型

方向3と7は，始点の描き出す方向と終点が一致せず，発散してしまいました。

方向1と5は，ひし型となってしまい，方向2，4，6は画面からはみ出してしまいました。

この場合適切な方向は0でしょう。

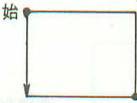
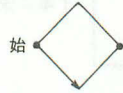
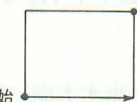
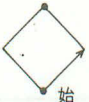
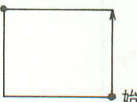
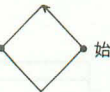
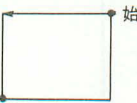
次に，箱型を描く場合の2点の位置関係による適切な方向を示しておきましょう。

2点の関係	結んだところ	方向
始	始	0
始	始	1
始	始	2
始	始	3
始	始	4
始	始	5
始	始	6
始	始	7

図4-4-4C　2点と箱型の適切な方向

4-4-5 円弧を描く　CIRCLE

N_{88}-BASIC (86) のCIRCLEに相当する命令ですが，GDCは，$\frac{1}{8}$円弧しか描くことができません。そのため，円を描くには，GBDSP (643 H) の描く方向を8回変えて，描かなければいけません。

4-1-1で説明を残しておいた，GBMDOT (65 AH，65 BH) マスキングドット数をここで説明致します。

円弧の描画方向の意味は，具体的には下図のようになっております。

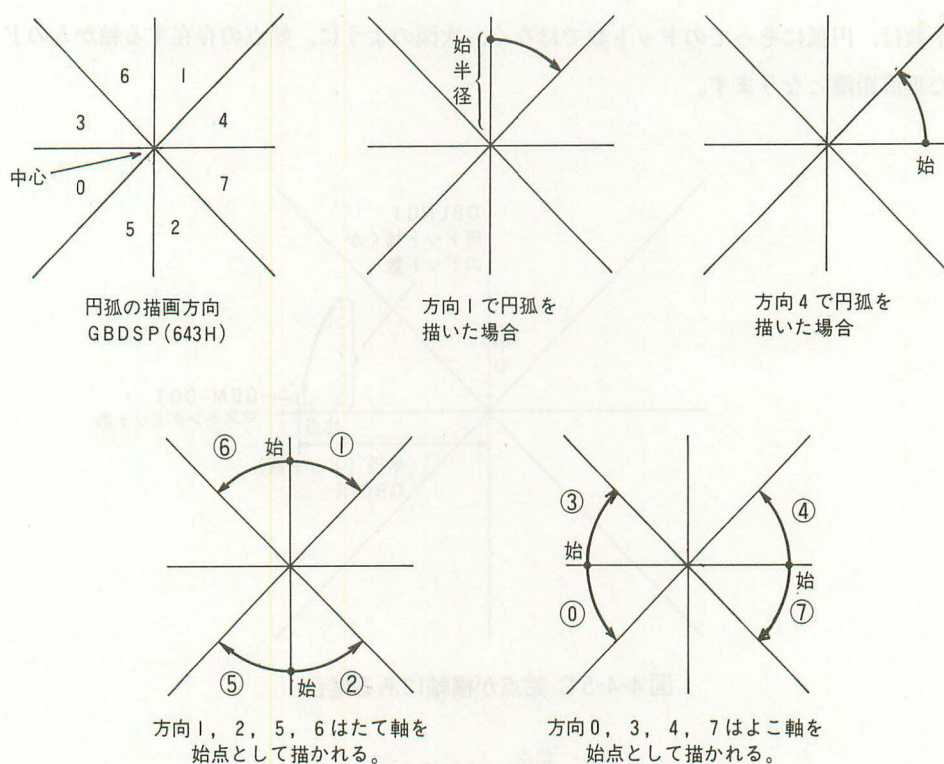
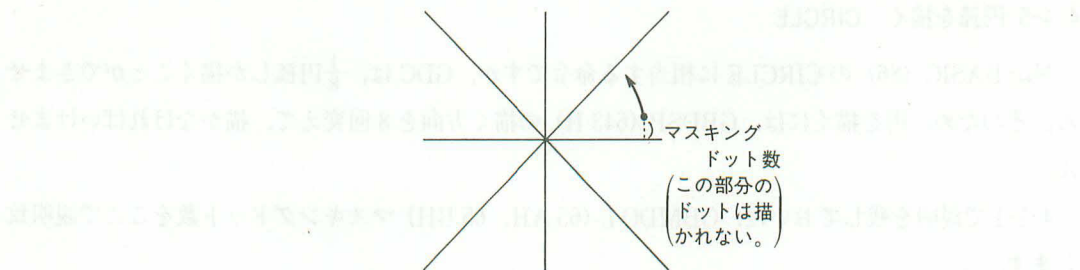



図4-4-5 A 円弧の描画方向の意味

ここで，マスキングドット数を指定すると，$\frac{1}{8}$円弧より少さな円弧を描くことができます。マスキングドット数を指定すると，そのドット数分だけ，始点からの円弧が描かれません（図4-4-5 B)。そのドット数分だけのドットが，マスクされるのです。

図 4-4-5 B　マスキングドット数の指定

ドット数は，円孤にそってのドット数ではなく，次図のように，始点の存在する軸からのドットで計った垂直距離となります。

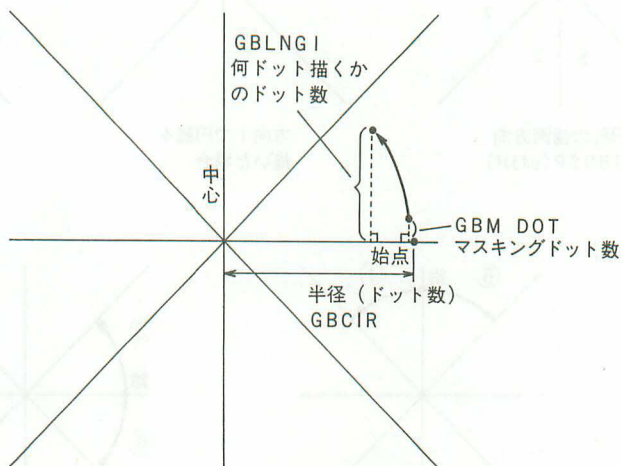



図 4-4-5 C　始点が横軸にある場合

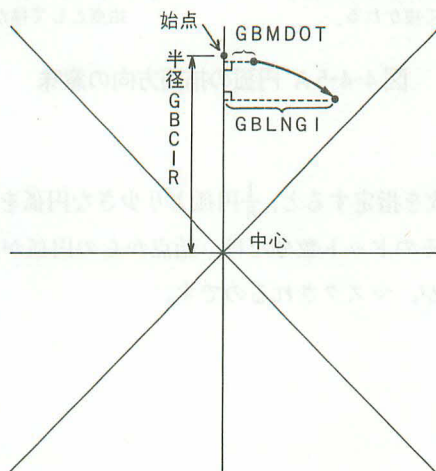



図 4-4-5 D　始点が縦軸にある場合

それでは，円孤描画の BIOS コールを説明しましょう。

AH レジスタに BIOS コード 48 H をセットします。

CH，DS，BX レジスタには，4-4-2 PSET と同じ値をセットします。

BIOS のワークエリアの中で表 4-4-5 のものを指定します。

GBON-PTN	６４０H	カラーパレット番号
GBDOTU	６４２H	単一画面のときのオペレーションモード
GBDSP	６４３H	描画開始方向
GBSX 1	６４８，９H	円孤の描画開始のＸ座標
GBSY 1	６４A，BH	円孤の描画開始のＹ座標
GBLNG 1	６４C，DH	描画ドット数
GBMDOT	６５A，BH	マスキングドット数
GBCIR	６５C，DH	円の半径（ドット数）
GBLPTN	６６０，１H	ラインスタイル
GBDTYP	６６８H	４（円孤）をセットする

表 4-4-5 円孤描画の BIOS ワークエリア

ただし，描画ドット数は，以下の理由により，制限があります。

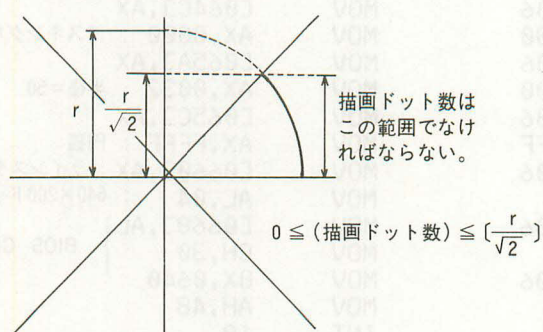



図 4-4-5 E 描画ドット数の制限

GBSX 1 と GBSY 1 は，円の中心ではなく，描画開始点を指定します。つまり，円の中心から半径のドット数だけずれた位置になります（図 4-4-5 F 参照)。

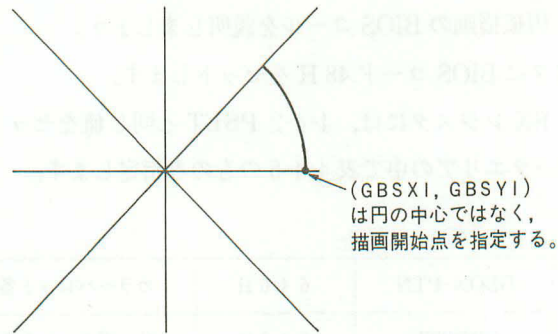



図 4-4-5 F 描画開始点の指定

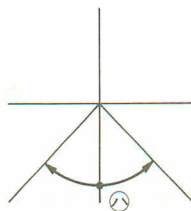
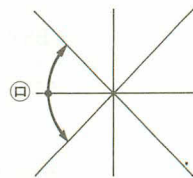
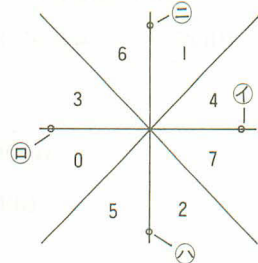
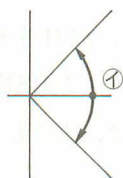
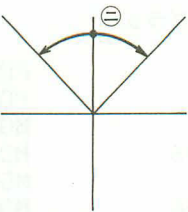
もう1つ注意することは，このコマンドで描画する円孤は専用高解像度（640×400）の縦横比で行われるということです。したがって，640×200 ドットモードで描画を行う場合は，縦方向に長い円孤となります。

円孤描画プログラム

```
0000 16         PUSH   SS   } DS=SS=60H
0001 1F         POP    DS   }
0002 B007       MOV    AL,07      ; カラーパレット=7
0004 A24006     MOV    [0640],AL
0007 B004       MOV    AL,04      ; 描画開始方向
0009 A24306     MOV    [0643],AL
000C B89001     MOV    AX,0190    ; 400+50=X
000F A34806     MOV    [0648],AX
0012 B86400     MOV    AX,0064    ; Y=100
0015 A34A06     MOV    [064A],AX
0018 B82400     MOV    AX,0024    ; 描画ドット数
001B A34C06     MOV    [064C],AX
001E B80000     MOV    AX,0000    ; マスキングなし
0021 A35A06     MOV    [065A],AX
0024 B83200     MOV    AX,0032    ; 半径=50
0027 A35C06     MOV    [065C],AX
002A B8FFFF     MOV    AX,FFFF    ; 円孤
002D A36006     MOV    [0660],AX  ; ラインスタイル=実線
0030 B004       MOV    AL,04      ; 640×200ドットモード
0032 A26806     MOV    [0668],AL }
0035 B530       MOV    CH,30     }  BIOS CALL
0037 BB4006     MOV    BX,0640   }
003A B448       MOV    AH,48
003C CD18       INT    18
003E 31C0       XOR    AX,AX
0040 CF         IRET
```

これを，描画開始点と方向を変えながら，8回くり返すと円ができます。次にそのサンプルプログラムを示します。描画開始点を㋑㋺㋩㋥と変えて描いています。

円の描画プログラム

```
0000 16           PUSH   SS
0001 1F           POP    DS
0002 B007         MOV    AL,07
0004 A24006       MOV    [0640],AL
0007 B004         MOV    AL,04          → 方向 4
0009 A24306       MOV    [0643],AL
000C B8C201       MOV    AX,01C2
000F A34806       MOV    [0648],AX
0012 B86400       MOV    AX,0064
0015 A34A06       MOV    [064A],AX
0018 B82500       MOV    AX,0025
001B A34C06       MOV    [064C],AX      この部分は前のサンプルプログラムと同じ。
001E B80000       MOV    AX,0000
0021 A35A06       MOV    [065A],AX
0024 B83200       MOV    AX,0032
0027 A35C06       MOV    [065C],AX
002A B8FFFF       MOV    AX,FFFF
002D A36006       MOV    [0660],AX
0030 B004         MOV    AL,04
0032 A26806       MOV    [0668],AL
0035 B530         MOV    CH,30
0037 BB4006       MOV    BX,0640
003A B448         MOV    AH,48
003C CD18         INT    18
003E B007         MOV    AL,07          ; 方向＝7
0040 A24306       MOV    [0643],AL
0043 CD18         INT    18
0045 A15C06       MOV    AX,[065C]      ; DX＝半径
0048 89C2         MOV    DX,AX
004A A14806       MOV    AX,[0648]      ; 描画開始点を㋺の位置にする
004D 29D0         SUB    AX,DX
004F 29D0         SUB    AX,DX
0051 A34806       MOV    [0648],AX
0054 B003         MOV    AL,03          ; 方向＝3
0056 A24306       MOV    [0643],AL
0059 B448         MOV    AH,48
005B CD18         INT    18
005D B000         MOV    AL,00          ; 方向＝0
005F A24306       MOV    [0643],AL
0062 CD18         INT    18
0064 A14806       MOV    AX,[0648]      ; 描画開始点を㋩の位置にする
0067 01D0         ADD    AX,DX
0069 A34806       MOV    [0648],AX
006C A14A06       MOV    AX,[064A]
006F 01D0         ADD    AX,DX
0071 A34A06       MOV    [064A],AX
0074 B002         MOV    AL,02          ; 方向＝2
0076 A24306       MOV    [0643],AL
0079 B448         MOV    AH,48
007B CD18         INT    18
007D B005         MOV    AL,05          ; 方向＝5
007F A24306       MOV    [0643],AL
0082 CD18         INT    18
0084 A14A06       MOV    AX,[064A]      ; 描画開始点を㊁の位置にする。
0087 29D0         SUB    AX,DX
0089 29D0         SUB    AX,DX
008B A34A06       MOV    [064A],AX
008E B001         MOV    AL,01      ; 方向＝1
```

```
0090 A24306          MOV     [0643],AL
0093 B448            MOV     AH,48
0095 CD18            INT     18
0097 B006            MOV     AL,06     ; 方向＝6
0099 A24306          MOV     [0643],AL
009C CD18            INT     18
009E 31C0            XOR     AX,AX
00A0 CF              IRET
```

4-4-6 グラフィックパターンを描く

基本パターンが8×8ドット以内で構成される，グラフィックパターンで，指定領域を埋めることができます。

AHレジスタにBIOSコード49Hをセットします。
CH，DS，BXレジスタを4-4-2 PSETと同じようにセットします。
BIOSワークエリアのうちで，次のものをセットします。

GBON-PTN	640H	カラーパレット
GBDOTU	642H	単一画面時 CHレジスタの b_5 b_4→11
GBDSP	643H	描画方向
GBSX 1	648，9H	描画開始X座標
GBSY 1	64A，BH	描画開始Y座標
GBLNG 1	64C，DH	よこ方向ドット数 (描画開始方向)
GBLNG 2	65E，FH	たて方向ドット数 (描画開始方向と垂直)
GBDOTI	660H～667H (8バイト)	グラフィック基本パターン格納バッファ。

表4-4-6 グラフィックパターンのBIOSワークエリア

次に，サンプルプログラムを示します。方向を0～7に変えたり，描画領域8×8を色々とかえてみると面白いですよ。

グラフィックパターン描画プログラム

```
0000 16          PUSH   SS  }
0001 1F          POP    DS  }      DS=SS=60H
0002 B007        MOV    AL,07     ; カラーパレット
0004 A24006      MOV    [0640],AL
0007 B002        MOV    AL,02     ; 方向
0009 A24306      MOV    [0643],AL
000C B8C800      MOV    AX,00C8   ; X=200
000F A34806      MOV    [0648],AX
0012 B86400      MOV    AX,0064   ; Y=100
0015 A34A06      MOV    [064A],AX
0018 B80800      MOV    AX,0008   ; 描画領域8×8
001B A34C06      MOV    [064C],AX
001E B80800      MOV    AX,0008
0021 A35E06      MOV    [065E],AX
0024 B90400      MOV    CX,0004   ; 基本パターンをBIOSのワークエリアに転送
0027 BE0001      MOV    SI,0100   ;
002A BF6006      MOV    DI,0660
002D 0E          PUSH   CS
002E 1F          POP    DS
002F 16          PUSH   SS
0030 07          POP    ES
0031 FC          CLD
0032 F3          REP
0033 A5          MOVSW
0034 16          PUSH   SS  }
0035 1F          POP    DS  }      DS=SS=60H
0036 BB4006      MOV    BX,0640
0039 B530        MOV    CH,30
003B B449        MOV    AH,49  }
003D CD18        INT    18     }   BIOS  CALL
003F 31C0        XOR    AX,AX
0041 CF          IRET
```

基本パターンを100Hからの8バイトに入れておきます。

次のパターンを入れて実行させて下さい。

```
0100 18 3C 7E DD FF 5A 81 42 00 00 00 00 00 00 00 00……パターン1
0110 18 3C 7E DD FF 24 5A A5 00 00 00 00 00 00 00 00……パターン2
0120 FF FF FF FF FF FF FF FF 00 00 00 00 00 00 00 00……パターン3
```

28H番地を10Hにすると，パターン2を表示できます。パターン3は消去用で，28H番地を20Hにすると表示できます。パターン3は，このままだとぬりつぶしになりますが，3H番地のカラーパレットNo.を0にすると，表示を消去することができます。

サンプルプログラムとこのデータを入力する時にDEF SEG=&H1F00として下さい。そのあとで次のプログラムを走らせてみて下さい。

インベーダーパターン描画プログラム

```
0 'SAVE"PAT2.BAS"
100 DEF USR=0
110 WHILE 1=1
120  POKE 3,0:POKE &H28,&H20:A=USR(0)
130  POKE 3,7:POKE &H28,&H10:A=USR(0):REM PATTERN 1
140  GOSUB *TIMER
150  POKE 3,0:POKE &H28,&H20:A=USR(0)
160  POKE 3,7:POKE &H28,0:A=USR(0)    :REM PATTERN 2
170  GOSUB *TIMER
180 WEND
190 END
200 *TIMER
210  FOR T=1 TO 500:NEXT
220 RETURN
```

例のインベーダーパターンが表示されたでしょう。

描画領域19Hと1FHをそれぞれ，0，50Hまたは，50H，0と変えて同じことを行って下さい。インベーダーが縦や横に一列に並んで表示されたでしょう。

N_{88}-BASIC (86) でいうならば，このパターンはPAINTのタイルパターンに相当するものです。

応用すれば，キャラクタ・ジェネレータにない文字などを自分で作って表示することもできます。

次にそのパターンの作成法を示します。

図 4-4-6 パターン作成法

8×8ドットのマス目を描き，図形を作成します。点のある位置を1，ない位置を0として，よこ8ビットを図4-4-6のように1バイトにまとめて数字化します。

4-4-7 高速書き込みモードにする

SCREENの第2パラメータの高速書き込みモードに相当します。高速書き込みモードには次の2種類あります。

① フラッシュモード…………画面がちらつくが②より高速

② フラッシュレスモード……画面はちらつかないで高速

SCREENの第2パラメータは①に相当します。

①はCHレジスタに06Hをセットし，②は16Hをセットし，単に次のBIOSコールをすればよいです。

```
MOV  AH, 4AH
INT  18H
```

SCREENの第2パラメータは，画面の一時停止（グラフィックマスクスイッチ）の機能もありますが，

```
MOV  AH, 41H
INT  18H
```

で，グラフィック画面は一時停止されます。再開するには，

```
MOV  AH, 40H
INT  18H
```

とします。

SCREENで高速書き込みをセレクトする場合は，画面がちらつきますので，

```
SCREEN, 3
```

で，高速書き込み，グラフィックをマスクしておいて，LINEなど，なんらかの処理をして，

```
SCREEN, 0
```

とするとよいでしょう。ただしこの場合，描画中は，グラフィック画面は消えます。表示したまま行う場合は，上記②のフラッシュレスモードで行うとよいでしょう。

4-5 マシン語によるG-VRAM直接アクセス法

G-VRAMは，GDCとCPUの二者でアクセスできるため，CPUから直接アクセスする場合は，GDCが描画していないときにアクセスしなければなりません。さもないと，期待した描画が行われないことがあります。

GDCが描画中かどうかを表わすフラグは，I/OポートA0Hをよみ，そのデータのb_3ビットを判断して，

1………描画中である。

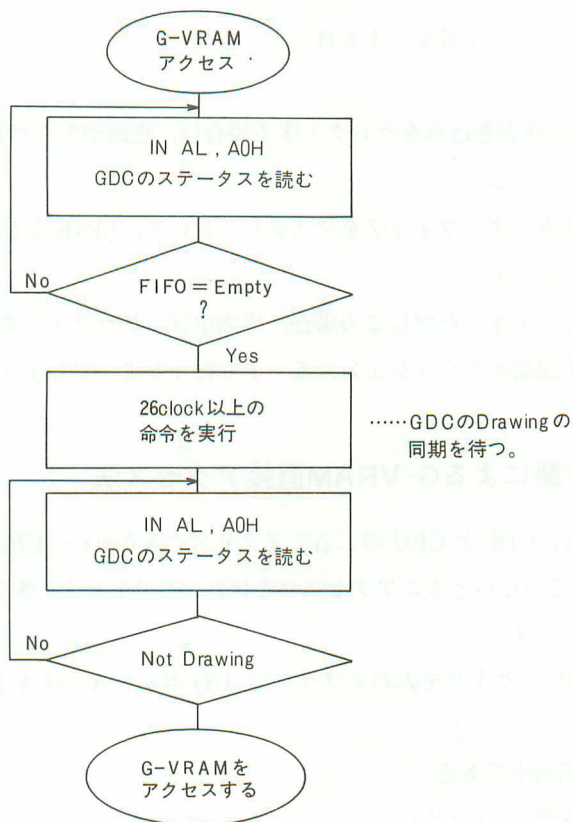
0………描画していない。

となります。

その他のビットの意味は，表4-5のようになります。

ビット	フラグ名称	機　　能
b_0	DATA Ready	GDCがReadなどの読み出しコマンド実行後読み出しデータが，読み出し可能になった。
b_1	FIFO Full	GDCのコマンドFIFOがいっぱいになった。
b_2	FIFO Empty	GDCのコマンドFIFOが空である。
b_3	Drawing	GDCが描画中。
b_4	DMA Execute	DMA転送を続行中である。
b_5	Vertical SYNC	垂直同期信号（VSYNC）が発生している。
b_6	Horizontal BLANC	水平消去信号（HBLANK）が発生している。
b_7	Light Pen Detect	ライトペン信号によるアドレスの検出が成された。

表4-5　I/OポートA0HのRead内容

フローチャートとサンプルプログラムを示しておきます。

G-VRAM直接アクセスのフローチャート

```
WAIT1： IN     AL，0A0H  ┐
        TEST   AL，04H   ├ FIFO Empty になるまでまつ。
        JE     WAIT1     ┘
        PUSH   AX        ┐
        PUSH   AX        │ 26クロック以上CPUを走らせる。
        POP    AX        ├ （VRAMアクセス前処理等
        POP    AX        ┘   をするとよい。）
WAIT2： IN     AL，0A0H  ┐
        TEST   AL，08H   ├ Not-Drawing になるまでまつ。
        JNE    WAIT2     ┘
          ↓
   ┌──────────────────┐
   │ VRAMをアクセスする │
   └──────────────────┘
```

G-VRAM 直接アクセスサンプルプログラム

それでは，G-VRAMを直接アクセスする実際のプログラムを1つ紹介します。これには，グラフィックの全画面を0.1秒でオールクリアするルーチンとランダムにクリアするルーチンとが入っています。ランダムクリアには次の3つのクリアの方法があります。

① カーテンコール・クリア

カーテンを開けるように画面の左から右へと段階的にクリアします。

② シャトル・クリア

画面の上下からラインが往復（シャトル）するようにクリアします。

③ バブル・クリア

水面上で泡が現われては消えていくようにジワジワとクリアします。

使い方は次のとおりです。

```
DEF SEG=&H1F00
HCLS=0
CALL HCLS（A%，B%）
```

A%＝0：B%＝0のときオールクリア
A%＝1のときランダムクリア
B%＝1……カーテンコールクリア
B%＝2……シャトルクリア
B%＝3……バブルクリア

```
1 'save "HCLS.BAS"
100 SCREEN 3,0
110 DEF SEG=&H1F00
120 HCLS=0
130 FOR I=0 TO &H11A
140  READ D$ : D=VAL("&H"+D$)
150  POKE I,D
160 NEXT I
170 LINE (0,0)-(639,399),6,BF
180 A%=0:B%=0 ' All clear
190 CALL HCLS(A%,B%)
200 LINE (0,0)-(639,399),7,BF
210 A%=1:B%=3 ' Bubble clear
220 CALL HCLS(A%,B%)
230 END
240 DATA C4,77,04,26,8B,04,8B,37,26,8B,1C,3D,00,00,74,06:'0000H
250 DATA 3D,01,00,74,2D,CF,E8,EF,00,B0,0C,E6,A2,B8,00,A8:'0010H
260 DATA E8,11,00,B8,00,B0,E8,0B,00,B8,00,B8,E8,05,00,B0:'0020H
270 DATA 0D,E6,A2,CF,B9,FF,3F,8E,C0,BF,00,00,33,C0,FC,F2:'0030H
280 DATA AB,C3,E8,C3,00,83,FB,01,74,0B,83,FB,02,74,44,83:'0040H
290 DATA FB,03,74,73,CF,B8,00,A8,E8,0D,00,B8,00,B0,E8,07:'0050H
300 DATA 00,B8,00,B8,E8,01,00,CF,B3,07,E8,0B,00,B3,03,E8:'0060H
310 DATA 06,00,B3,00,E8,01,00,C3,8E,D8,33,FF,B9,50,00,51:'0070H
320 DATA 57,B9,90,01,88,1D,83,C7,50,E0,F9,5F,83,C7,01,59:'0080H
330 DATA E0,ED,C3,B8,00,A8,E8,0D,00,B8,00,B0,E8,07,00,B8:'0090H
340 DATA 00,B8,E8,01,00,CF,8E,D8,33,DB,B9,FF,7C,E8,12,00:'00A0H
350 DATA 87,CB,E8,0D,00,87,CB,43,43,49,49,8A,C5,3C,7F,75:'00B0H
360 DATA EC,C3,C6,47,01,00,C3,B9,F5,1E,B8,00,A8,E8,0D,00:'00C0H
370 DATA B8,00,B0,E8,07,00,B8,00,B8,E8,01,00,CF,8E,D8,33:'00D0H
380 DATA DB,33,D2,51,D0,FD,D0,D9,72,07,D0,EE,D0,DA,E9,F3:'00E0H
390 DATA FF,59,52,8A,C7,0C,00,8A,F8,C6,47,01,00,03,D9,5A:'00F0H
400 DATA 4A,8A,C6,0A,C2,75,EB,C3,E4,A0,A8,04,74,FA,E4,A0:'0100H
410 DATA A8,20,74,FA,E4,A0,A8,08,75,FA,C3,00,00,00,00,00:'0110H
```

4-6 グラフィックLIOとBASICコマンド

4-4でグラフィックBIOSを紹介しましたが，GDCを直接コントロールするコマンドばかりでした。

PC-8001やPC-8801は，BASICコマンドの処理エントリーアドレスを使って，マシン語から直接BASICのステートメントを実行できました。PC-9801は，この考え方を進めて，BASICのステートメントのメイン処理エントリーをINT命令でコールする形式でまとめたLIO（Logical Input Output）というものが，ROMの中にプログラムされています。

グラフィックLIOは，

　ＩＮＴ　Ａ０Ｈ～ＩＮＴ　ＡＦＨ

　ＩＮＴ　ＣＥＨ

となっています。

ここで，BASICのステートメントとLIO，BIOS，I/O等のハードウェアの構造を図4-6としてまとめておきましょう。

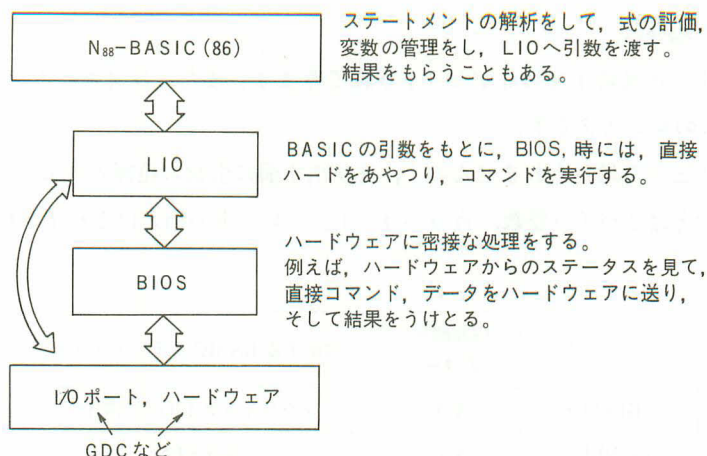



図 4-6 LIO の位置

では，これから，グラフィックに関する LIO の使い方を BASIC のステートメントと対応づけながら説明していきましょう。

4-6-1 引数の渡し方

引数は，テキストセグメント 60 H の，オフセットアドレス 150 AH から，以下の図のように設定します。引数は，BASIC のステートメントの引数と対応しています。INT コールするときは，DS に 60 H，BX に 150 AH を設定しておきます。

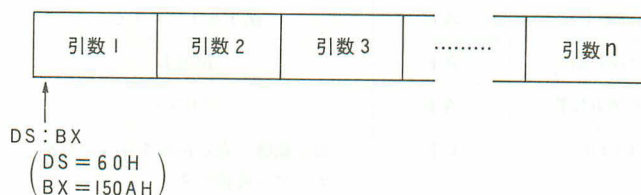



図 4-6-1 グラフィック LIO のパラメータの渡し方

（正常終了時は，AH レジスタに 0 がセットされてきます。）

4-6-2 グラフィック LIO コマンド一覧

グラフィック LIO のコマンド一覧表（表 4-6-2）を示します。個々のコマンドの説明は，4-6-3 から行います。

特に指定がない場合は，

AHレジスタに正常終了なら0がセットされてきます。また，保証されるレジスタは，DS，SS，SPの3つのレジスタです。

引数の範囲のチェックは行われませんので，呼び出す前に引数の範囲のチェックを行って下さい。

X，Y座標の指定は2バイト整数，カラーコード，パレットの指定は1バイト整数です。

コマンド名	内部割り込みコード	対応するBASICステートメント
GINIT	A 0	グラフィックLIOの初期化
GSCREEN	A 1	SCREEN
GVIEW	A 2	VIEW
GCOLOR 1	A 3	COLOR
GCOLOR 2	A 4	COLOR=（パレット#，カラー#）
GCLS	A 5	CLS
GPSET	A 6	PSET／PRESET
GLINE	A 7	LINE
GCIRCLE	A 8	CIRCLE
GPAINT 1	A 9	PAINT 色でぬりつぶす
GPAINT 2	A A	PAINT タイルパターン
GGET	A B	GET
GPUT 1	A C	画をPUTする
GPUT 2	A D	漢字をPUTする
GROLL	A E	ROLL
GPOINT	A F	POINT
GCOPY	C E	指定領域の表示情報を指定の格納エリアへ移動する

表4-6-2 グラフィックLIOコマンド一覧

4-6-3 GINIT (INT A 0 H)

(1) 機　能

グラフィック画面，パレット番号等の初期化を行います。このコマンドを実行すると，

① 画面モードはカラーグラフィックモード

② アクティブ画面は0

③ ディスプレイ画面は1

④ パレット番号と対応する，同じカラーコードがセットされる。即ち,カラーパレットの初期化。

⑤　アクティブ画面全体は初期状態になります。

⑵　入力条件

●ＩＮＴ　Ａ０Ｈで呼び出します。

●ＤＳ⇐６０Ｈ

●引数はありません。

⑶　出力条件

●保障されるレジスタ……ＤＳ，ＳＳ，ＳＰ

●ＡＨが0なら正常終了

4-6-4 GSCREEN　(INT　A1H)

⑴　機　　能

画面モード，画面スイッチ，アクティブ画面，ディスプレイ画面をセットします。

⑵　引数テーブル

BX＝150AH（DS＝60H）
⇩

＋0	＋1	＋2	＋3
画面モード	画面スイッチ	アクティブ画面	ディスプレイ画面

それぞれの設定値は，次表のようになります。

画面モード

設定値	モード
0	カラーグラフィック640×200
1	モノクログラフィック640×200
2	640×400モノクロ
3	640×400カラー
ＦＦＨ	今までのモード変更しない

画面スイッチ

設定値	グラフィック画面表示	高速書き込み
0	有	しない
1	有	する
2	無	する
3	無	する
ＦＦＨ	今までのモード変更しない	

アクティブ画面

画面モードの設定値	画面モードの設定値に対する指定範囲	←プレーン番号−1に相当する
0	0～1	2ページ
1	0～5	6ページ
2	0～2	3ページ
3	0	1ページ

ディスプレイ画面・プレーンについては，図4-1-1を参照して下さい。

① 640×200カラー

設定値	表示画面
0	表示しない
1	プレーン1を表示
2	プレーン2を表示

② 640×200モノクロ

設定値	表示画面
0	表示しない
1	プレーン1
2	プレーン2
3	プレーン1と2合成
4	プレーン3
5	プレーン1と3合成
6	プレーン2と3合成
7	プレーン1,2,3合成
8	表示しない
9	プレーン4
A	プレーン5
B	プレーン4と5
C	プレーン6
D	プレーン4と6
E	プレーン5と6
F	プレーン4,5,6

③ 640×400モノクロ

設定値	表示画面
0, 8	表示しない
1	プレーン1
2	プレーン2
3	プレーン1と2
4	プレーン3
5	プレーン1と3
6	プレーン2と3
7	プレーン1,2,3

④ 640×400カラー

設定値	表示画面
0, 8	表示しない
1	プレーン1

4-6-5 GVIEW (INT A2H)

(1) 機　能

アクティブ画面内のビューポートを指定します。また，ビューポート内のぬりつぶし，外枠の描画を行います。このコマンド発行後図形描画は，ビューポート内にのみ行われます。

(2) 引数テーブル

BX＝150AH (DS＝60H)
⇩

0, 1	2, 3	4, 5	6, 7	8	9
X1	Y1	X2	Y2	領域色	境界色

BASICのステートメントの

VIEW (X1,Y1)－(X2,Y2), <領域色>, <境界色> に相当する。

ただし，X1,Y1,X2,Y2は，2バイトの符号付整数値です。

ただし，実際に表示が有効な領域は，

$0 \leqq X \leqq 639$

0≦Y≦１９９（６４０×２００）

0≦Y≦３９９（６４０×４００）

です。

領域色と境界色はカラーパレット番号の０～７を設定します。変更しない場合は，FFH とします。

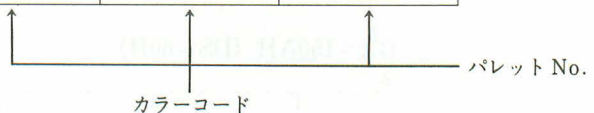
4-6-6 GCOLOR 1　(INT　A 3 H)

(1) 機　　能

バックグラウンドカラー，ボーダーカラー，フォアグラウンドカラーを指定します。

① バックグラウンドカラーとは，グラフィック画面の地の色のことで，この命令実行後 CLS 命令によって画面をクリアすると，この色によって画面がぬり変えられます。以後 PRESET 命令を指定なしで実行すると，この色が採用されます。

② ボーダーカラーとは，グラフ LIO が制御可能なディスプレイ画面の外側の色のことです。ただし，専用高解像度ディスプレイ装置接続時は意味がありません。

③ フォアグラウンド･カラーとは，図形描画においてパレット番号省略時に使用される色のことです。

(2) 引数リスト

変更しない場合は，FFH を入れます。

［１］COLOR ステートメントの〈ファンクションコード〉の設定を除いたものに相当する。ステートメントの引数〈ファンクションコード〉に対応する部分が未使用になっています。

4-6-7 GCOLOR 2　(INT　A 4 H)

N_{88}-BASIC (86) のステートメント

　　COLOR＝(〈パレット番号〉，〈カラーコード〉)

に相当します。

引数リスト

BX＝150AH (DS＝60H)

⇩

＋0	＋1
パレット番号	カラーコード

4-6-8 GCLS　(INT　A 5 H)

N_{88}-BASIC (86) のステートメント

　　　CLS　2

に相当します。

引数リストはありません。

4-6-9 GPSET　(INT　A 6 H)

N_{88}-BASIC (86) のステートメント

　　PSET (X，Y)，CまたはPRESET (X，Y) に色Cの点をうつ，に相当します。

引数リスト

BX＝150AH (DS＝60H)

⇩

＋0，1	＋2，3	＋4
X	Y	パレット番号

↑ 前の色と同じにするときはFFHにする。

ただし，

　AH………1　　PSET

　　　　　2　　PRESET

PSET，PRESETの指定をAHレジスタで行います。

4-6-10 GLINE　(INT　A７H)

N₈₈-BASIC (86) のステートメント

ＬＩＮＥ (Ｘ１, Ｙ１)－ (Ｘ２, Ｙ２), 〈パレットNo.〉, | Ｂ / ＢＦ |, 〈ラインスタイル〉

に相当します。

引数リスト

BX＝150AH (DS＝60H)

＋0, 1	＋2, 3	＋4, 5	＋6, 7	＋8	＋9	＋AH	＋BH	＋CH
							ラインスタイル	
X1	Y1	X2	Y2	パレット No.	描画コード	ラインスタイルスイッチ	Lo	Hi

4-6-11 GCIRCLE　(INT　A８H)

N₈₈-BASIC (86) のCIRCLEステートメントに相当します。BASICでの引数の指定の仕方は，次のとおりです。

ＣＩＲＣＬＥ (Ｘ, Ｙ), 〈半径〉, 〈パレットNo.〉, 〈開始角度〉, 〈終了角度〉, 〈比率〉

↑ 中心点

GCIRCLEコマンドの引数の指定の仕方は，少し異ります。

引数リスト

BX＝150AH (DS＝60H)

＋0, 1	＋2, 3	＋4, 5	＋6, 7	＋8	＋9	＋A, B	＋C, D	＋E, F	＋10, 11
CX	CY	RX	RY	パレット No.	各フラグ	SX	SY	EX	EY

BX＋9　各フラグの説明（各ビットに意味がある）

b_0………0　　開始点指示が無

　　　　　1　　開始点指示が有

b_1………0　　開始点線分指示が無

　　　　　1　　開始点線分指示が有

b_2………0　　終了点指示が無

　　　　　1　　終了点指示が有

b_3………0　　終了点線分指示が無

　　　　　1　　終了点線分指示が有

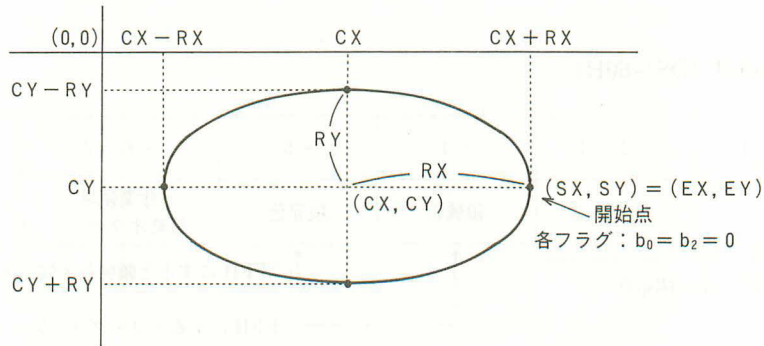
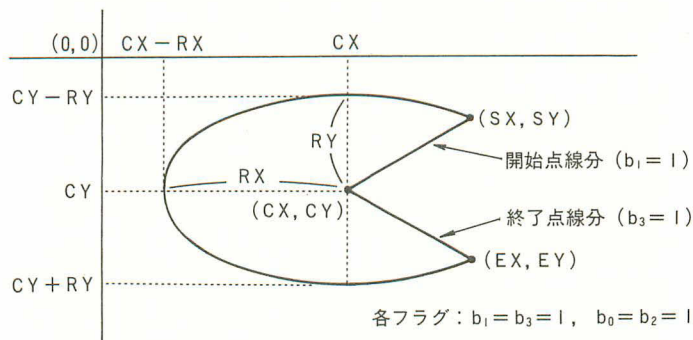
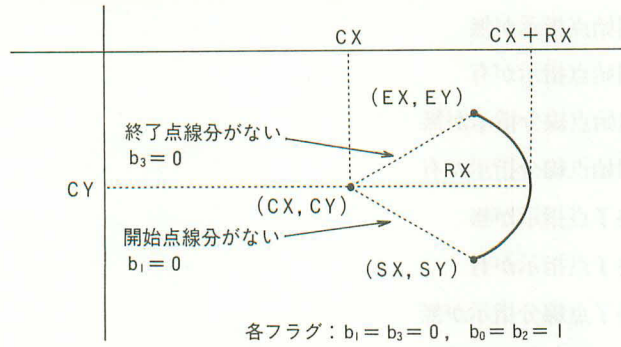
b_4………開始点・終了点一致時に，(SX,SY)＝(EX,EY)

　　　0………全楕円

　　　1………一致点のみ描画

b_5～b_7　　未使用

開始点線分，終了点線分とは，それぞれ中心点と開始点，終了点を結ぶ線のことです。

描画図形と各引数の対応図を示します。

4-6-12 GPAINT 1　(INT　A 9 H)

指定された境界色で囲まれた領域を，指定された色でペイントします。N_{88}-BASIC (86) の

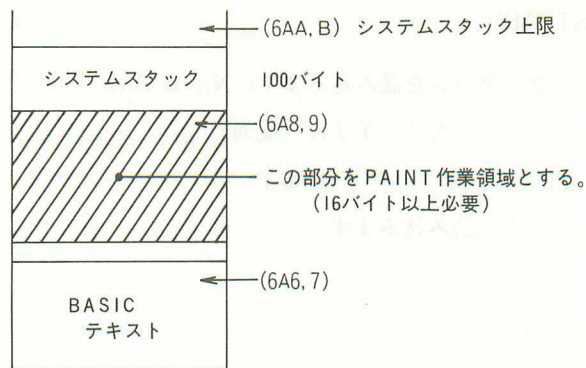
PAINT（X，Y)，〈領域色〉，〈境界色〉

に相当します。

引数リスト

BX＝150AH（DS＝60H)
⇩

＋0，1	＋2，3	＋4	＋5	＋6，7	＋8，9
X	Y	領域色	境界色	作業領域 最終オフセット	作業領域 TOP オフセット

作業領域は，BASIC テキストのあとのフリーエリアにとるとよいでしょう。

作業領域不足の場合は，AH レジスタに，07H を入れて中断します。

作業領域は必ずテキストセグメント（60 H)になければなりません。

4-6-13 GPAINT 2　(INT　AAH)

指定した点と境界色で決定される領域を，指定のタイル・パターンでぬりつぶします。

N_{88}-BASIC（86）のステートメント

PAINT（X，Y），〈タイルストリング〉，〈境界色〉，〈バックグラウンドストリング〉

↑ N_{88}-BASIC（86）では意味がありません。無視します。

に相当します。

引数リスト

BX＝150AH（DS＝60H）
⇩

＋0，1	＋2，3	＋4	＋5	＋6，7（タイルパターン）	＋8，9（タイルパターン）	＋AH	＋B～＋FH	＋10，11H	＋12，13H
X	Y	未使用		オフセットアドレス	セグメントアドレス	境界色	未使用	作業域 END オフセット	作業域 TOP オフセット

X, Y：ぬりつぶし開始点
＋5：タイルパターン長

作業域については，GPAINT 1 と同じです。

BX＋5 タイルパターン長は，タイルパターン格納領域の大きさをバイト単位で指定します。モノクロの場合 1～FFH，カラーの場合 3～FFH です。

タイルパターン自体は別のセグメントにあってもよく，オフセットアドレス（BX+6,7）とセグメントアドレス（BX+8,9）で指定します。

4-6-14 GGET　(INT ABH)

画面上のグラフィックパターンを読み込みます。N_{88}-BASIC (86) の

GET@(X 1, Y 1) − (X 2, Y 2), 〈配列〉

に相当します。

配列ではなくメモリー上へ読み込みます。

引数リスト

BX＝150AH (DS＝60H)
⇩

+0, 1	+2, 3	+4, 5	+6, 7	+8, 9	+A, B	+C, D
X 1	Y 1	X 2	Y 2	格納領域		格納領域の長さ(バイト)
				オフセット	セグメント	

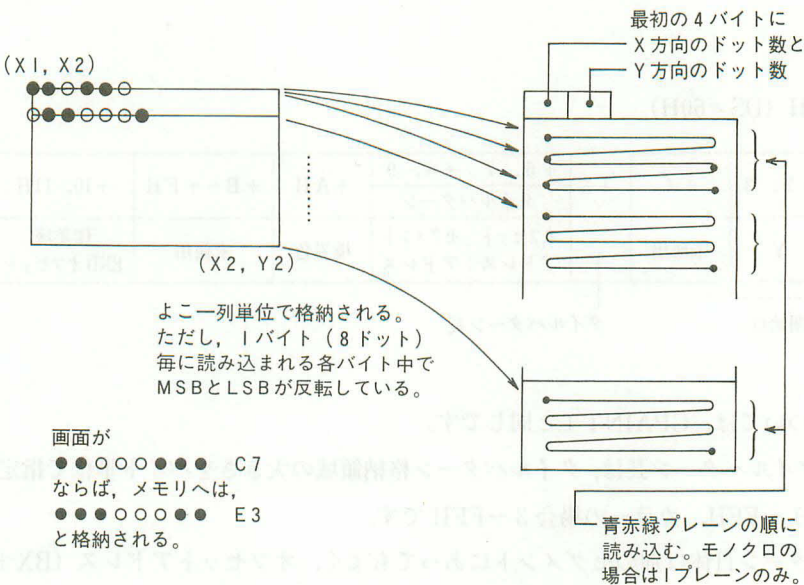
必要な格納領域の長さの計算方法は,

カラーの場合

{(X 2 − X 1 + 8) ¥ 8} ∗ (Y 2 − Y 1 + 1) ∗ 3 + 4

モノクロの場合

{(X 2 − X 1 + 8) ¥ 8} ∗ (Y 2 − Y 1 + 1) + 4

です。

格納領域の形式

4-6-15 GPUT 1　(INT　ACH)

GGET の逆を行います。

N_{88}-BASIC (86) の

PUT＠(X，Y)，〈配列変数名〉，〈条件〉，〈フォアグラウンドカラー〉，〈バックグラウンドカラー〉

に相当します。

配列変数が GGET で用いた格納領域になります。

引数リスト

BX＝150AH (DS＝60H)

⇩

＋0，1	＋2，3	＋4，5	＋6，7	＋8，9	＋A	＋B	＋C	＋D
X	Y	格納領域		格納領域の長さ	描画モード条件	カラーSW	フォアグラウンドカラー	バックグラウンドカラー
		オフセットアドレス	セグメントアドレス					

（＋0，1〜＋8，9）GGET の格納領域と同じです。

（＋B カラーSW）
0……指定なし。
1……モノクロで色をつけて表示。

格納されている画面が白黒の場合，BX＋0 BH のカラーSW を 1 にしますと，白 (1) のドットを BX＋0 CH のフォアグラウンドカラーで，黒 (0) のドットを BX＋0 DH のバックグラウンドカラーで表示することができます。

BX＋0 AH 描画モードというのは，条件に対応するもので，次の表のように BASIC の条件と対応しています。

描画モード	条 件
0	PSET
1	PRESET
2	OR
3	AND
4	XOR

4-6-16 GPUT 2　(INT　ADH)

漢字を画面に表示するもので，

PUT＠(X，Y)，KANJI(〈漢字コード〉)，〈条件〉，〈フォアグラウンドカラー〉，〈バックグラウンドカラー〉

に対応します。

引数リスト

BX＝150AH (DS＝60H)
⇩

＋0，1	＋2，3	＋4，5	＋6	＋7	＋8	＋9
X	Y	漢字コード	描画モード条件	カラーSW	フォアグラウンドカラー	バックグラウンドカラー

GPUT 1と同じ

4-6-17 GROLL (INT AEH)

N_{88}-BASIC (86) のステートメント

ROLL

と同等のコマンドです。

引数リスト

BX＝150AH (DS＝60H)
⇩

＋0，1
ドット数

↑

上方向へ移動するドット数
640×200のとき C8H (200) 未満
640×400のとき 190H (400) 未満
でなければならない。

4-6-18 GPOINT (INT AFH)

N_{88}-BASIC (86) のステートメント

POINT (X, Y)

に相当します。

引数リスト

BX＝150AH（DS＝60H）
⇩

＋0，1	＋2，3
X	Y

出力情報は，AL レジスタに入れられて戻ります。

ALの値	出力情報の意味
FFH	指定座標がアクティブ画面のビューポート以外である。
0～7H	画面モードがカラーの場合，指定座標のパレット番号を示す。
0～1H	画面モードがモノクロの場合，0…黒，1…白を示す。

4-6-19 GCOPY　(INT　CEH)

現在 CRT 画面に表示されているディスプレイ画面上の指定領域におけるドット状態を指定の格納領域へ複写します。

したがって，N_{88}-BASIC（86）の COPY ステートメントとは全く異質のものであります。

引数リストは，4-6-1～4-6-18 のように引数テーブルを RAM 上に設けるのではなく，次のようにレジスタ渡しとなります。

レジスタ　　　内　容

AX←指定領域左上点X座標（X）

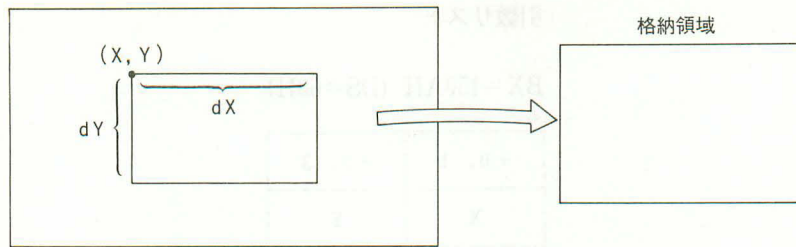
BX←指定領域左上点Y座標（Y）

CL←指定領域X方向ドット数（dX）

CH←指定領域Y方向ドット数（dY）

DI←格納領域オフセットアドレス

ES←格納領域セグメントアドレス

次図を参照して下さい。

ただし，

○X，dXは8の倍数でなければなりません。

○X＋dX＜280H（640）

○640×200のとき　　Y≦C7H（199）

　　　　　　　　　　Y＋dY－1≦C7H（199）

○640×400のとき　　Y≦18FH（399）

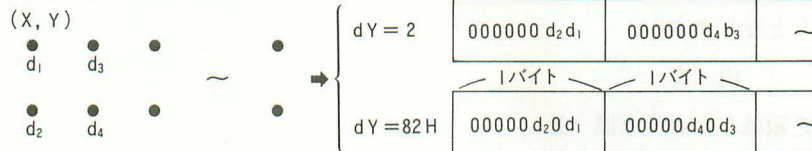
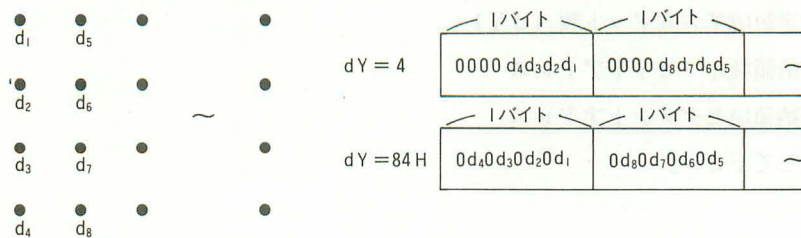
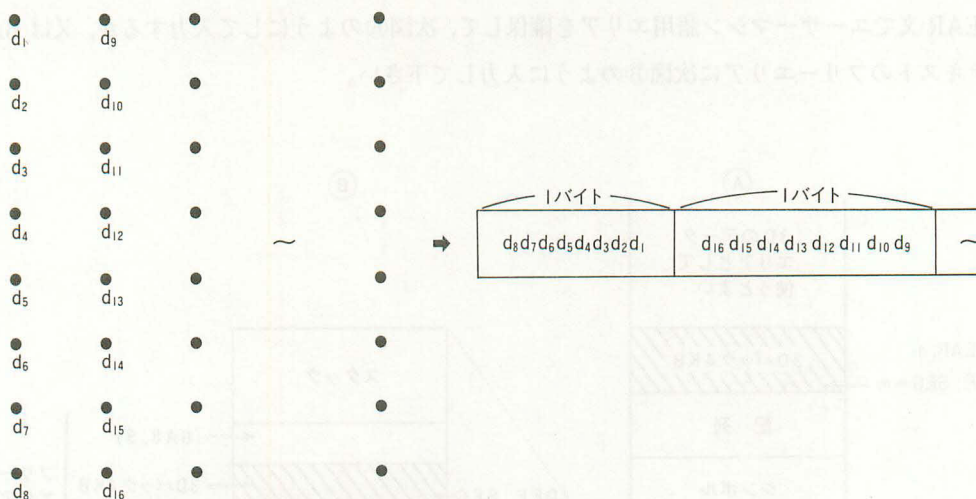
　　　　　　　　　　Y＋dY－1≦18FH（399）

○dXは2（82Hでもよい），4（84H），8のいずれかです。

格納のされ方は，画面合成を考慮に入れて，表示中のドットが，

　カラーの場合　　0以外

　モノクロの場合　白のとき

格納するドットdiを1とします。

4-7 3Dパッケージの紹介

この章のしめくくりとして，マシン語で書いた3Dパッケージを紹介しましょう。

3Dパッケージというのは，

「3 Dimensional package」

の略で内容は，3次元座標で与えられたデータを，透視法で見た2次元図形のデータに変換するプログラムのパッケージです。

プログラムは，大きく3つに分けられます。

① 3次元図形データを2次元図形データに変換するプログラム。

② ①の出力の2次元図形データを画面にプロットするプログラム。

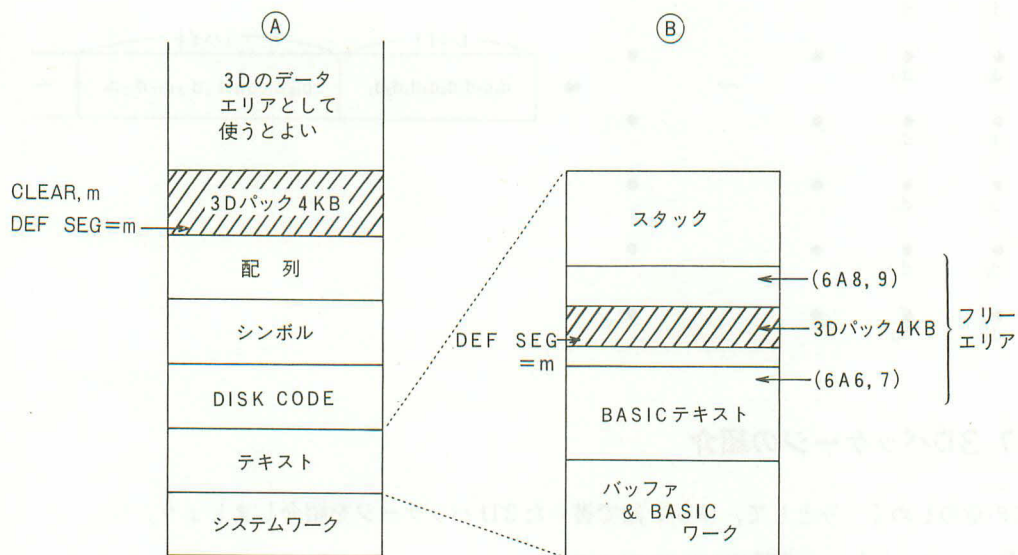
③ 画面をクリアするプログラム。

の3つです。

マシン語レベルのCALL命令で呼び出した場合は，RET命令で戻ればよいですが，BASICレベルのUSRやCALLで呼び出した場合は，IRET命令で戻らねばなりません。そのために，3つのエントリーアドレステーブルを用意しました。

上記①②③のプログラムのエントリーは，次のようになっています。

	INT呼び出し用（BASIC）	セグメント内コール CALL用	セグメント間コール CALLF用
①	80H	A0H	C0H
②	83H	A3H	C3H
③	86H	A6H	C6H

プログラムは，全体でオフセットアドレス0000 H～0 FFFH の約4 Kバイトありますので，CLEAR 文でユーザーマシン語用エリアを確保して，次図Ⓐのようにして入力するか，又はBASICのテキストのフリーエリアに次図Ⓑのように入力して下さい。

ただし，Ⓑの場合は，BASIC テキストのエンドポインタ(6 A 6,7)とスタックの下限のポインタ(6 A 8,9) の内容をみて，その間に入るようにしなければなりません。

少なくとも，

(6 A 8，9)－(6 A 6，7)≧1 0 0 0 H　←3 Dパッケージの大きさ

をみたしていなければ，Ⓑの方法は使えません。また，Ⓑの方法は，PAINT 文と同時には使えません。

3 Dパッケージのワークエリアは，3 Dパッケージをロードしたセグメントアドレスのオフセット0～7 FHです。ユーザーが値をセットすべきワークエリアは，次のようになっています。

名　称	オフセットアドレス	意　　　　　味
UPER	0000H	0のとき　プレーン1～3 1のとき　プレーン4～6　をセレクトする。
COLOR	0001H	クリアするプレーンNo. 1………1（4） 2………2（5） 4………3（6） 同時にクリアするときは，値を足したものにする。 （例）プレーン1と3を同時にクリアするとき，UPER＝0にして， 1＋4＝5をCOLORにセットする。
X-WIDTH	0002，3H	画面のX方向の幅の$\frac{1}{2}$を入れる。
Y-WIDTH	0004，5H	画面のY方向の幅の$\frac{1}{2}$を入れる。

平行移動1

DX1	0006，7H	X方向の平行移動量
DY1	0008，9H	Y方向の平行移動量
DZ1	000A，BH	Z方向の平行移動量

平行移動2

DX2	000C，DH	X方向の平行移動量
DY2	000E，FH	Y方向の平行移動量
DZ2	0010，11H	Z方向の平行移動量

回転移動

PITCH	0012H	ピッチ方向の回転量
BANK	0013H	バンク方向の回転量
HEADING	0014H	ヘディング方向の回転量

回転量は，符号付1バイトの整数である。

　－128～127

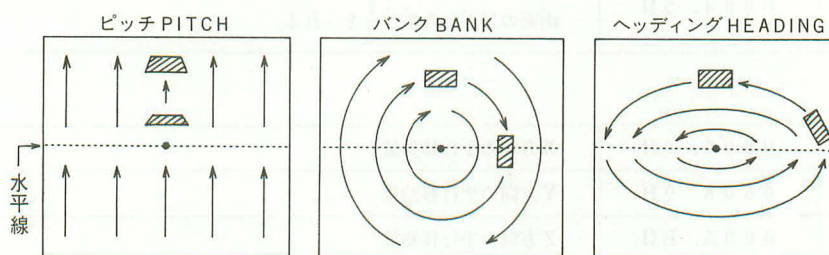
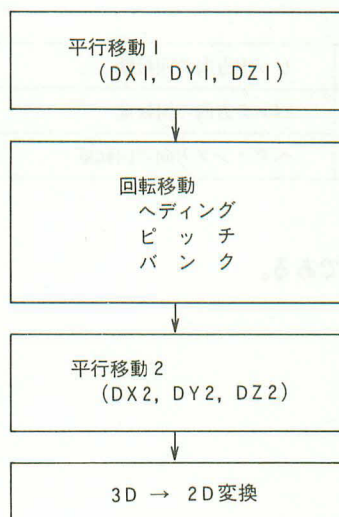
を表わせるが，

　－180～179°

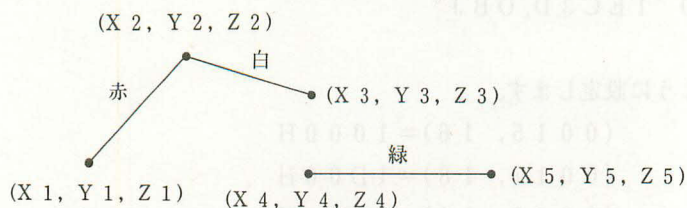
に対応している。

　1≒1.4°

に対応している。

IBP	0015，16H	Input Buffer Pointer　入力となる3次元データの先頭オフセットアドレス
	0017，18H	同じく，セグメントアドレス
OBP	0019，1AH	Output Buffer Pointer　出力となる2次元データの先頭オフセットアドレス
	001B，1CH	同じく，セグメントアドレス

変換の順番は，下図の順番で行われます。

3次元入力データの形式は，先頭の1バイトに，出発点か中継点かを示すコード(上位4ビット)と色コード(下位4ビットの内3ビット)をおき，次の6バイトに，2バイトずつX，Y，Z座標値をおくという形式で入力します。データの終わりは，点属性・色コードに当たる部分にFFHを入れて下さい。

出発点は５，中継点は６です。

線は１本１本に色をつけることができます。 $\left(\begin{array}{c}0\sim7\\ \text{黒}\sim\text{白}\end{array}\right)$

５２Ｈ ｝１バイト　←— ＩＢＰ

Ｘ１ ……２バイト
Ｙ１ ……２バイト
Ｚ１ ……２バイト

上位，下位を逆に入れてはいけない。これは，サブロジック社やその他の３Ｄパッケージのデータと互換性をもたせるためである。
符号付２バイトの整数である。
−32768〜32767を表わすことができる。

６２Ｈ …… 赤 …… ２
Ｘ２
Ｙ２
Ｚ２
６７Ｈ …… 白 …… ７
Ｘ３
Ｙ３
Ｚ３
５４Ｈ …… 出発点５，緑 …… ４
Ｘ４
Ｙ４
Ｚ４
６４Ｈ …… 中継点６，緑 …… ４
Ｘ５
Ｙ５
Ｚ５
ＦＦＨ …… データの終了点（エンドマーク）

３次元入力データの入っている先頭アドレスのオフセットとセグメントをIBP（0015，16Ｈ……オフセット，0017，18Ｈ……セグメント）にセットして下さい。

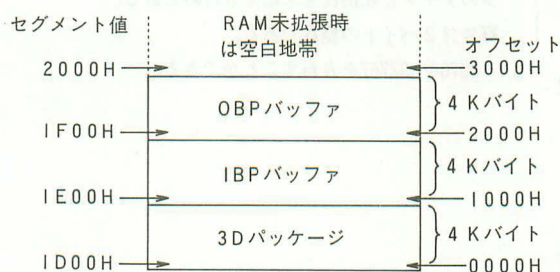
OBPには，出力２次元データの入いるべきバッファの先頭オフセットとセグメントを入れて下さい。

次に，セグメントとIBP・OBPバッファの設定例を示します。

```
CLEAR ,&H1D00
OK
DEF SEG=&H1D00
OK
BLOAD "TEC3D.OBJ"
OK
```

そして，以下のように設定します。

```
IBP     (0015, 16)=1000H
        (0017, 18)=1D00H
OBP     (0019, 1A)=2000H
        (001B, 1C)=1D00H
```

図で示すと次のように領域をとったことになります。

したがって，RAMを増設すれば，IBPバッファ，OBPバッファはそれぞれ64Kバイトまで設定することができます。3Dから2Dに変換するとデータ量が減りますから，IBPバッファを64Kバイトにとっても OBPバッファは64Kバイト未満となるでしょう。

実際の3Dパッケージプログラムを次ページからにダンプリストとしてのせておきます。セグメントを前述の設定例のようにとって，モニタのEコマンドで入力してゆくとよいでしょう。入力し終わったら，一度ディスクに

```
BSAVE "TEC3D. OBJ",0,&H1000
```

とBSAVEしておきます。

次に示すチェックサムプログラムをうち込み，RUNをし，

```
           SEGMENT ADDRESS=1D00
     START OFFSET ADDRESS=0
     END   OFFSET ADDRESS=E70
OUTPUT TO P)rinter or C)RT =C
```

と入力して，チェックサムが合っているか確認して下さい。もし合ってないときは，データが違っているのですから，もう一度その行を調べ訂正しチェックサムが一致するまで，これをくり返して下さい。最後の

OUT PUT TO P)rinter or C)RT=

に対して，Pと答えるとプリンターに表示されます。

チェックサムプログラム

```
0 'SAVE"CHECKSUM"
100 '
110 '   CHECK SUM FOR PC-9800
120 '
130  CRLF$=CHR$(13)+CHR$(10)
140  DEF FNHXB$(X)=RIGHT$("0"+HEX$(X),2)
150  DEF FNHXSM$(X)=RIGHT$("00"+HEX$(X),3)
160  DEF FNHXW$(X)=RIGHT$("000"+HEX$(X),4)
170 '
180  INPUT "              SEGMENT ADDRESS=",SG$:SG=VAL("&H"+SG$):DEF SEG=SG
190  INPUT "         START OFFSET ADDRESS=",S$:S=VAL("&H"+S$)
200  INPUT "         END   OFFSET ADDRESS=",E$:E=VAL("&H"+E$)
210  INPUT " OUTPUT TO P)rinter or C)RT =",O$:O$=CHR$(ASC(O$) AND &HDF)
220  IF O$="P" THEN OPEN "LPT1:" FOR OUTPUT AS #1
230  IF O$="C" THEN OPEN "SCRN:" FOR OUTPUT AS #1
240 '
250  FOR I=S TO E
260   IF CN MOD 256=0 THEN P$=CRLF$+CRLF$+
      " ADD : +0  *  *  *  *  *  * +7  +8  *  *  *  *  *  * +F :: SUM"
      :GOSUB *PRN
270   IF CN MOD 16 =0 THEN P$=CRLF$+FNHXW$(I)+" :":GOSUB *PRN
280   IF CN MOD 16=8 THEN P$=" ":GOSUB *PRN
290   DA=PEEK(I):P$=" "+FNHXB$(DA):GOSUB *PRN
300   SUM=SUM+DA
310   IF CN MOD 16=15 THEN P$=" :: "+FNHXSM$(SUM):GOSUB *PRN:SUM=0
320  CN=CN+1:NEXT
330  END
340 '
350 *PRN
360  PRINT #1,P$;
370 RETURN
```

3Dパッケージ・ダンプリスト（チェックサム付）

```
 ADD : +0  *  *  *  *  *  * +7  +8  *  *  *  *  *  * +F :: SUM
0000 : 00 07 7D 00 64 00 00 00  00 00 00 00 00 00 00 00 :: 0E8
0010 : 00 00 00 00 00 00 10 00  1D 00 20 00 1D 00 00 00 :: 06A
0020 : 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 :: 000
0030 : 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 :: 000
0040 : 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 0A :: 00A
0050 : 00 0A 00 64 00 64 00 00  00 00 00 00 00 00 00 00 :: 0D2
0060 : 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 :: 000
0070 : 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 :: 000
0080 : E9 7D 00 E9 92 00 E9 A7  00 00 00 00 00 00 00 00 :: 471
0090 : 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 :: 000
00A0 : E9 65 00 E9 7A 00 E9 8F  00 00 00 00 00 00 00 00 :: 429
00B0 : 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 :: 000
00C0 : E9 4D 00 E9 62 00 E9 77  00 00 00 00 00 00 00 00 :: 3E1
00D0 : 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 :: 000
00E0 : 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 :: 000
00F0 : 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 :: 000

 ADD : +0  *  *  *  *  *  * +7  +8  *  *  *  *  *  * +F :: SUM
0100 : 0E 1F E8 3B 01 33 C0 CF  1E 0E 1F E8 32 01 1F C3 :: 55B
0110 : 1E 0E 1F E8 2A 01 1F CB  0E 1F E8 15 0D 33 C0 CF :: 541
0120 : 1E 0E 1F E8 0C 0D 1F C3  1E 0E 1F E8 04 0D 1F CB :: 45C
0130 : 0E 1F E8 93 0C 33 C0 CF  1E 0E 1F E8 8A 0C 1F C3 :: 621
0140 : 1E 0E 1F E8 82 0C 1F CB  32 F6 22 FF 79 05 80 F6 :: 6E8
0150 : 80 F7 DB 87 CB 22 FF 79  05 80 F6 80 F7 DB 52 8B :: 9E8
0160 : D1 E8 08 00 5A 22 F6 79  02 F7 DB C3 33 C9 2B DA :: 844
0170 : 73 03 03 DA 49 41 D1 E3  D1 E1 2B DA 73 03 03 DA :: 79B
0180 : 49 41 D1 E3 D1 E1 2B DA  73 03 03 DA 49 41 D1 E3 :: 886
0190 : D1 E1 2B DA 73 03 03 DA  49 41 D1 E3 D1 E1 2B DA :: 8FF
01A0 : 73 03 03 DA 49 41 D1 E3  D1 E1 2B DA 73 03 03 DA :: 79B
01B0 : 49 41 D1 E3 D1 E1 2B DA  73 03 03 DA 49 41 D1 E3 :: 886
01C0 : D1 E1 2B DA 73 03 03 DA  49 41 D1 E3 D1 E1 2B DA :: 8FF
01D0 : 73 03 03 DA 49 41 D1 E3  D1 E1 2B DA 73 03 03 DA :: 79B
01E0 : 49 41 D1 E3 D1 E1 2B DA  73 03 03 DA 49 41 D1 E3 :: 886
01F0 : D1 E1 2B DA 73 03 03 DA  49 41 D1 E3 D1 E1 2B DA :: 8FF

 ADD : +0  *  *  *  *  *  * +7  +8  *  *  *  *  *  * +F :: SUM
0200 : 73 03 03 DA 49 41 D1 E3  D1 E1 2B DA 73 03 03 DA :: 79B
0210 : 49 41 D1 E3 D1 E1 2B DA  73 03 03 DA 49 41 D1 E3 :: 886
0220 : 87 CB C3 98 8B D8 03 DB  81 C3 20 0B 8B 0F 86 E9 :: 866
0230 : 04 40 98 8B D8 03 DB 81  C3 20 0B 8B 07 86 E0 C3 :: 747
0240 : FC C4 06 15 00 A3 41 00  8C 06 43 00 C4 06 19 00 :: 477
0250 : A3 45 00 8C 06 47 00 A0  14 00 E8 C6 FF A3 23 00 :: 5E8
0260 : 89 0E 1D 00 A0 12 00 E8  B9 FF A3 25 00 89 0E 1F :: 584
0270 : 00 A0 13 00 E8 AC FF A3  27 00 89 0E 21 00 8B 0E :: 561
0280 : 23 00 A1 27 00 F7 E9 D0  E4 D1 D2 D0 E4 D1 D2 52 :: 9CB
0290 : 8B 0E 1D 00 A1 1F 00 F7  E9 D0 E4 D1 D2 D0 E4 D1 :: 932
02A0 : D2 A1 21 00 F7 EA D0 E4  D1 D2 D0 E4 D1 D2 59 03 :: A7F
02B0 : D1 89 16 2F 00 8B 0E 23  00 A1 21 00 F7 E9 D0 E4 :: 6B1
02C0 : D1 D2 D0 E4 D1 D2 8B DA  52 8B 0E 1D 00 A1 1F 00 :: 827
02D0 : F7 E9 D0 E4 D1 D2 D0 E4  D1 D2 A1 27 00 F7 E9 D0 :: C06
02E0 : E4 D1 D2 D0 E4 D1 D2 59  2B D1 89 16 31 00 8B 0E :: 89C
02F0 : 1D 00 A1 25 00 F7 E9 D0  E4 D1 D2 D0 E4 D1 D2 89 :: 9FA
```

```
 ADD : +0  *  *  *  *  *  * +7  +8  *  *  *  *  *  * +F :: SUM
0300 : 16 33 00 8B 0E 25 00 A1  21 00 F7 E9 D0 E4 D1 D2 :: 700
0310 : D0 E4 D1 D2 89 16 35 00  8B 0E 25 00 A1 27 00 F7 :: 6A8
0320 : E9 D0 E4 D1 D2 D0 E4 D1  D2 89 16 37 00 A1 1F 00 :: 92D
0330 : F7 D8 A3 39 00 8B 0E 1D  00 A1 27 00 F7 E9 D0 E4 :: 7BD
0340 : D1 D2 D0 E4 D1 D2 52 8B  0E 23 00 A1 1F 00 F7 E9 :: 8A8
0350 : D0 E4 D1 D2 D0 E4 D1 D2  A1 21 00 F7 EA D0 E4 D1 :: BD6
0360 : D2 D0 E4 D1 D2 59 2B D1  89 16 3B 00 8B 0E 1D 00 :: 70E
0370 : A1 21 00 F7 E9 D0 E4 D1  D2 D0 E4 D1 D2 52 8B 0E :: A3B
0380 : 23 00 A1 1F 00 F7 E9 D0  E4 D1 D2 D0 E4 D1 D2 A1 :: A12
0390 : 27 00 F7 E9 D0 E4 D1 D2  D0 E4 D1 D2 59 03 D1 89 :: A6B
03A0 : 16 3D 00 8B 0E 23 00 A1  25 00 F7 E9 D0 E4 D1 D2 :: 70C
03B0 : D0 E4 D1 D2 89 16 3F 00  8B 36 41 00 C4 3E 45 00 :: 67E
03C0 : 8B 2E 43 00 8C C0 95 8E  C0 26 8A 04 46 A2 01 00 :: 5C8
03D0 : 24 F0 3C 50 75 11 E8 89  00 89 16 1D 00 89 0E 1F :: 509
03E0 : 00 89 1E 21 00 EB E2 3C  60 74 09 8C C0 95 8E C0 :: 6DD
03F0 : 32 C0 AA C3 E8 6B 00 89  16 23 00 89 16 29 00 89 :: 5C5
```

```
 ADD : +0  *  *  *  *  *  * +7  +8  *  *  *  *  *  * +F :: SUM
0400 : 0E 25 00 89 0E 2B 00 89  1E 27 00 89 1E 2D 00 E8 :: 37F
0410 : 5E 01 73 39 8C C0 95 8E  C0 A0 01 00 AA 8B 0E 1D :: 63B
0420 : 00 8B 1E 21 00 E8 3F 01  AA 8B 0E 1F 00 8B 1E 21 :: 41E
0430 : 00 E8 19 01 AA 8B 0E 23  00 8B 1E 27 00 E8 27 01 :: 448
0440 : AA 8B 0E 25 00 8B 1E 27  00 E8 01 01 AA A1 29 00 :: 496
0450 : A3 1D 00 A1 2B 00 A3 1F  00 A1 2D 00 A3 21 00 E9 :: 4C9
0460 : 62 FF 8B 04 46 46 86 E0  03 06 06 00 A3 29 00 8B :: 548
0470 : 04 46 46 86 E0 03 06 08  00 A3 2B 00 8B 04 46 46 :: 3F0
0480 : 86 E0 03 06 0A 00 A3 2D  00 8B 0E 29 00 A1 2F 00 :: 3DB
0490 : F7 E9 D0 E4 D1 D2 D0 E4  D1 D2 52 8B 0E 2B 00 A1 :: A45
04A0 : 35 00 F7 E9 D0 E4 D1 D2  D0 E4 D1 D2 59 03 D1 52 :: A42
04B0 : 8B 0E 2D 00 A1 3B 00 F7  E9 D0 E4 D1 D2 D0 E4 D1 :: 95E
04C0 : D2 59 03 D1 52 8B 0E 29  00 A1 31 00 F7 E9 D0 E4 :: 779
04D0 : D1 D2 D0 E4 D1 D2 52 8B  0E 2B 00 A1 37 00 F7 E9 :: 8C8
04E0 : D0 E4 D1 D2 D0 E4 D1 D2  59 03 D1 52 8B 0E 2D 00 :: 8F3
04F0 : A1 3D 00 F7 E9 D0 E4 D1  D2 D0 E4 D1 D2 59 03 D1 :: A99
```

```
 ADD : +0  *  *  *  *  *  * +7  +8  *  *  *  *  *  * +F :: SUM
0500 : 52 8B 0E 29 00 A1 33 00  F7 E9 D0 E4 D1 D2 D0 E4 :: 8D3
0510 : D1 D2 52 8B 0E 2B 00 A1  39 00 F7 E9 D0 E4 D1 D2 :: 8CA
0520 : D0 E4 D1 D2 59 03 D1 52  8B 0E 2D 00 A1 3F 00 F7 :: 773
0530 : E9 D0 E4 D1 D2 D0 E4 D1  D2 8B DA 59 03 D9 59 5A :: AE4
0540 : 03 16 0C 00 03 0E 0E 00  03 1E 10 00 C3 E8 F8 FB :: 413
0550 : 8B 0E 04 00 51 8B C3 F7  E9 D0 E4 D1 D2 D0 E4 D1 :: 9F8
0560 : D2 59 03 D1 8A C2 C3 E8  DE FB 8B 0E 02 00 EB E4 :: 939
0570 : 32 F6 E8 FA 00 72 02 B6  01 E8 01 01 72 03 80 CE :: 6E2
0580 : 02 E8 07 01 72 03 80 CE  04 E8 0D 01 72 03 80 CE :: 572
0590 : 08 32 D2 8B 0E 23 00 8B  1E 27 00 03 D9 80 FF 80 :: 573
05A0 : 72 02 B2 01 8B 1E 27 00  2B D9 80 FF 80 72 03 80 :: 5EF
05B0 : CA 02 8B 0E 25 00 8B 1E  27 00 2B D9 80 FF 80 72 :: 5CF
05C0 : 03 80 CA 04 8B 1E 27 00  03 D9 80 FF 80 72 03 80 :: 5F1
05D0 : CA 08 8A C4 22 C2 74 01  C3 8A C6 0A C2 F9 75 01 :: 7C7
05E0 : C3 3A D6 73 02 86 D6 8A  C6 B1 04 D2 C0 0A C2 B9 :: 8C0
05F0 : 21 00 8B DF BF 0C 06 0E  07 F2 AE 8B FB 74 02 F8 :: 705
```

```
ADD  : +0  *  *  *  *  *  * +7  +8  *  *  *  *  *  * +F :: SUM
0600 : C3 BB 2D 06 03 D9 03 D9  8B 0F FF E1 1E 2D 4B 5A :: 6D3
0610 : 4A 1A 69 49 29 16 25 24  14 12 78 68 58 48 28 18 :: 384
0620 : 0F 0E 0D 0B 0A 09 08 07  06 05 04 02 01 AC 08 C8 :: 1E5
0630 : 08 74 08 D3 08 ED 08 6C  09 90 08 E0 08 FA 08 4F :: 59A
0640 : 09 8F 09 72 09 94 09 15  09 31 09 47 09 7F 08 95 :: 37D
0650 : 08 94 09 3F 09 07 09 23  09 CD 08 9B 08 B7 08 79 :: 3D9
0660 : 08 34 08 B1 08 63 08 05  08 94 09 94 09 94 09 8B :: 3D7
0670 : 0E 1D 00 8B 1E 21 00 03  D9 80 FF 80 C3 8B 0E 1D :: 549
0680 : 00 8B 1E 21 00 2B D9 80  FF 80 C3 8B 0E 1F 00 8B :: 5D3
0690 : 1E 21 00 2B D9 80 FF 80  C3 8B 0E 1F 00 8B 1E 21 :: 587
06A0 : 00 03 D9 80 FF 80 C3 A1  1D 00 8B 1E 23 00 89 1E :: 5CF
06B0 : 1D 00 A3 23 00 A1 1F 00  8B 1E 25 00 89 1E 1F 00 :: 337
06C0 : A3 25 00 A1 21 00 8B 1E  27 00 89 1E 21 00 A3 27 :: 3EC
06D0 : 00 C3 73 03 E8 D0 FF A1  23 00 A3 49 00 A1 25 00 :: 666
06E0 : A3 4B 00 8B 0E 27 00 89  0E 4D 00 C3 3B D9 74 01 :: 4DE
06F0 : C3 5B 5B 5B 53 B9 12 04  3B D9 74 01 5B 22 C0 C3 :: 67F

ADD  : +0  *  *  *  *  *  * +7  +8  *  *  *  *  *  * +F :: SUM
0700 : 2B D9 80 FF 80 F5 D1 DB  03 D9 C3 E8 6F FF E8 C1 :: B42
0710 : FF 8B 1E 23 00 E8 D4 FF  8B 0E 4B 00 8B 1E 1F 00 :: 632
0720 : E8 DD FF 53 8B 0E 4D 00  8B 1E 21 00 E8 D1 FF 53 :: 7D2
0730 : 8B 0E 49 00 8B 1E 1D 00  E8 C5 FF 59 3B D9 75 0E :: 644
0740 : 89 1E 1D 00 5B 89 1E 1F  00 89 0E 21 00 C3 53 2B :: 3DE
0750 : D9 80 FF 80 5B 72 0E 89  1E 49 00 58 A3 4B 00 89 :: 672
0760 : 0E 4D 00 EB B3 89 1E 1D  00 58 A3 1F 00 89 0E 21 :: 48F
0770 : 00 EB A5 F7 1E 1D 00 F7  1E 23 00 E8 8D FF F7 1E :: 783
0780 : 1D 00 F7 1E 23 00 C3 E8  01 FF E8 45 FF 8B 1E 25 :: 6FA
0790 : 00 E8 58 FF 8B 0E 49 00  8B 1E 1D 00 E8 61 FF 53 :: 682
07A0 : 8B 0E 4D 00 8B 1E 21 00  E8 55 FF 53 8B 0E 4B 00 :: 523
07B0 : 8B 1E 1F 00 E8 49 FF 59  3B D9 75 0E 89 1E 1F 00 :: 5AE
07C0 : 5B 89 1E 1D 00 89 0E 21  00 C3 53 2B D9 80 FF 80 :: 5F0
07D0 : 5B 72 0F 89 1E 4B 00 5B  89 1E 49 00 89 0E 4D 00 :: 3FD
07E0 : EB B2 89 1E 1F 00 5B 89  1E 1D 00 89 0E 21 00 EB :: 525
07F0 : A3 F7 1E 1F 00 F7 1E 25  00 E8 8B FF F7 1E 1F 00 :: 6B7

ADD  : +0  *  *  *  *  *  * +7  +8  *  *  *  *  *  * +F :: SUM
0800 : F7 1E 25 00 C3 E8 7F FF  8B 1E 21 00 4B 80 FF 80 :: 777
0810 : 72 01 C3 E8 67 FE 72 01  C3 E8 53 FE 72 03 E8 52 :: 8A1
0820 : FF E8 75 FE 72 01 C3 E8  C7 FF E8 50 FE 72 03 E8 :: AD1
0830 : D9 FE F9 C3 E8 D4 FE 8B  1E 21 00 4B 80 FF 80 72 :: 9D3
0840 : 01 C3 E8 54 FE 72 01 C3  E8 40 FE 72 03 E8 37 FF :: 8ED
0850 : E8 1C FE 72 01 C3 E8 1A  FF E8 3D FE 72 03 E8 90 :: 949
0860 : FF F9 C3 E8 A5 FE E8 22  FE 72 01 C3 E8 2A FE 72 :: B06
0870 : 03 E8 7D FF E8 10 FF F9  C3 E8 F7 FE E9 E7 FF E8 :: CAE
0880 : F1 FE E8 14 FE 72 01 C3  E8 00 FE 72 03 E8 F7 FE :: A57
0890 : E8 5E FF F9 C3 E8 73 FE  E9 E7 FF E8 E9 FE E8 CE :: DAE
08A0 : FD 72 01 C3 E8 D6 FD 72  03 E8 5F FE E8 C4 FE F9 :: B4B
08B0 : C3 E8 3D FF E9 E7 FF E8  37 FF E8 C0 FD 72 01 C3 :: BAF
08C0 : E8 AC FD 72 03 E8 AB FE  E8 40 FE F9 C3 E8 B7 FE :: C16
08D0 : E9 E7 FF E8 B1 FE E8 96  FD 72 03 E8 95 FE F9 C3 :: C8D
08E0 : E8 0E FF E8 89 FD 72 03  E8 88 FE F9 C3 E8 97 FE :: B7F
08F0 : E8 8A FD 72 03 E8 13 FE  F9 C3 E8 F4 FE E8 7D FD :: BD5
```

```
 ADD : +0  *  *  *  *  *  * +7  +8  *  *  *  *  *  * +F :: SUM
0900 : 72 03 E8 06 FE F9 C3 E8  7D FE E8 62 FD 72 01 C3 :: 9FD
0910 : E8 60 FE F9 C3 E8 D9 FE  E8 54 FD 72 01 C3 E8 52 :: B6A
0920 : FE F9 C3 E8 61 FE E8 54  FD 72 01 C3 E8 DC FD F9 :: C2A
0930 : C3 E8 BD FE E8 46 FD 72  01 C3 E8 CE FD F9 C3 E8 :: C1E
0940 : C9 FD E8 2E FE F9 C3 E8  3D FE E8 A4 FE F9 C3 E8 :: CE7
0950 : 9F FE A0 22 00 22 C0 79  01 C3 E8 20 FD 72 03 E8 :: 7E0
0960 : A9 FD E8 0A FD 72 03 E8  09 FE F9 C3 E8 18 FE E9 :: A9C
0970 : E0 FF E8 96 FD A0 22 00  22 C0 79 01 C3 E8 0B FD :: 92B
0980 : 72 03 E8 02 FE E8 11 FD  72 03 E8 64 FE F9 C3 E8 :: 9B6
0990 : E1 FD EB E1 8B 0E 27 00  8B 1E 21 00 2B D9 80 FF :: 7B7
09A0 : 80 73 03 E8 01 FD A1 27  00 48 48 80 FC 80 72 01 :: 6A3
09B0 : C3 A1 23 00 A3 49 00 A1  25 00 A3 4B 00 A1 27 00 :: 4EF
09C0 : A3 4D 00 8B 0E 1D 00 8B  1E 49 00 E8 32 FD 53 8B :: 58D
09D0 : 0E 1F 00 8B 1E 4B 00 E8  26 FD 53 8B 0E 21 00 8B :: 4C4
09E0 : 1E 4D 00 E8 1A FD 4B 75  12 43 89 1E 21 00 5B 89 :: 52B
09F0 : 1E 1F 00 5B 89 1E 1D 00  E9 75 FB 8A C7 43 22 C0 :: 62B

 ADD : +0  *  *  *  *  *  * +7  +8  *  *  *  *  *  * +F :: SUM
0A00 : 78 02 EB 0E 89 1E 21 00  58 A3 1F 00 58 A3 1D 00 :: 46D
0A10 : EB B1 89 1E 4D 00 58 A3  4B 00 58 A3 49 00 EB A3 :: 6A8
0A20 : 00 00 FE 6D FC DC FB 4A  F9 BA F8 2A F6 9B F5 0E :: 9F1
0A30 : F3 83 F1 FA F0 73 EE EE  ED 6B EB EC EA 70 E8 F7 :: D08
0A40 : E7 82 E6 10 E4 A2 E3 39  E1 D4 E0 74 DF 18 DD C2 :: AA0
0A50 : DC 71 DB 25 D9 E0 D8 A0  D7 66 D6 32 D5 05 D3 DE :: A4E
0A60 : D2 BE D1 A5 D0 94 CF 89  CE 86 CD 8B CC 98 CB AC :: B49
0A70 : CA C9 C9 ED C9 1A C8 50  C7 8E C6 D5 C6 25 C5 7D :: A61
0A80 : C4 DF C4 49 C3 BD C3 3A  C2 C1 C2 51 C1 EB C1 8E :: ABE
0A90 : C1 3A C0 F1 C0 B1 C0 7B  C0 4E C0 2C C0 13 C0 04 :: 8E9
0AA0 : C0 00 C0 04 C0 13 C0 2C  C0 4E C0 7B C0 B1 C0 F1 :: 8AE
0AB0 : C1 3A C1 8E C1 EB C2 51  C2 C1 C3 3A C3 BD C4 49 :: A16
0AC0 : C4 DF C5 7D C6 25 C6 D5  C7 8E C8 50 C9 1A C9 ED :: A71
0AD0 : CA C9 CB AC CC 98 CD 8B  CE 86 CF 89 D0 94 D1 A5 :: B4C
0AE0 : D2 BE D3 DE D5 05 D6 32  D7 66 D8 A0 D9 E0 DB 25 :: A91
0AF0 : DC 71 DD C2 DF 18 E0 74  E1 D4 E3 39 E4 A2 E6 10 :: A84

 ADD : +0  *  *  *  *  *  * +7  +8  *  *  *  *  *  * +F :: SUM
0B00 : E7 82 E8 F7 EA 70 EB EC  ED 6B EE EE F0 73 F1 FA :: CFB
0B10 : F3 83 F5 0E F6 9B F8 2A  F9 BA FB 4A FC DC FE 6D :: B67
0B20 : 00 00 01 92 03 23 04 B5  06 45 07 D5 09 64 0A F1 :: 401
0B30 : 0C 7C 0E 05 0F 8C 11 11  12 94 14 13 15 8F 17 08 :: 2E8
0B40 : 18 7D 19 EF 1B 5D 1C C6  1E 2B 1F 8B 20 E7 22 3D :: 550
0B50 : 23 8E 24 DA 26 1F 27 5F  28 99 29 CD 2A FA 2C 21 :: 5A2
0B60 : 2D 41 2E 5A 2F 6B 30 76  31 79 32 74 33 67 34 53 :: 4A7
0B70 : 35 36 36 12 36 E5 37 AF  38 71 39 2A 39 DA 3A 82 :: 58F
0B80 : 3B 20 3B B6 3C 42 3C C5  3D 3E 3D AE 3E 14 3E 71 :: 532
0B90 : 3E C5 3F 0E 3F 4E 3F 84  3F B1 3F D3 3F EC 3F FB :: 707
0BA0 : 40 00 3F FB 3F EC 3F D3  3F B1 3F 84 3F 4E 3F 0E :: 644
0BB0 : 3E C5 3E 71 3E 14 3D AE  3D 3E 3C C5 3C 42 3B B6 :: 5DA
0BC0 : 3B 20 3A 82 39 DA 39 2A  38 71 37 AF 36 E5 36 12 :: 57F
0BD0 : 35 36 34 53 33 67 32 74  31 79 30 76 2F 6B 2E 5A :: 4A4
0BE0 : 2D 41 2C 21 2A FA 29 CD  28 99 27 5F 26 1F 24 DA :: 55F
0BF0 : 23 8E 22 3D 20 E7 1F 8B  1E 2B 1C C6 1B 5D 19 EF :: 56C
```

```
 ADD : +0  *  *  *  *  *  * +7   +8  *  *  *  *  *  * +F :: SUM
0C00 : 18 7D 17 08 15 8F 14 13   12 94 11 11 0F 8C 0E 05 :: 2F5
0C10 : 0C 7C 0A F1 09 64 07 D5   06 45 04 B5 03 23 01 92 :: 489
0C20 : 7C 80 67 D2 F9 DC AF 3D   92 84 3C C3 FC DC 7D 81 :: 9E1
0C30 : 6F D2 07 DD AF 3D 92 85   3C C3 0A DD E8 92 00 E8 :: 870
0C40 : 9D 00 E8 A0 00 A0 01 00   A8 01 74 20 BF 00 40 A0 :: 5A2
0C50 : 00 00 D0 E8 73 04 81 C7   40 1F E8 B9 00 E8 D7 00 :: 736
0C60 : E8 F5 00 E8 30 01 E8 59   01 A0 01 00 A8 02 74 20 :: 617
0C70 : BF 00 80 A0 00 00 D0 E8   73 04 81 C7 40 1F E8 95 :: 732
0C80 : 00 E8 B3 00 E8 D1 00 E8   0C 01 E8 35 01 A0 01 00 :: 608
0C90 : A8 04 74 1D BF 00 C0 A0   00 00 D0 E8 73 04 81 C7 :: 6D3
0CA0 : 40 1F E8 71 00 E8 8F 00   E8 AD 00 E8 E8 00 E8 11 :: 78D
0CB0 : 01 C3 50 E4 A0 24 02 75   FA 58 E6 A2 C3 50 E4 A0 :: 8A4
0CC0 : 24 02 75 FA 58 E6 A0 C3   E8 F2 FF 8A C4 E8 ED FF :: B31
0CD0 : C3 B0 78 E8 DC FF B0 FF   E8 E2 FF E8 DF FF C3 B0 :: D5F
0CE0 : 20 E8 CE FF C3 8B 1E 4F   00 8B 16 53 00 A1 51 00 :: 676
0CF0 : 8B 0E 55 00 3B DA 74 03   73 18 C3 3B C1 74 FB 72 :: 6A5

 ADD : +0  *  *  *  *  *  * +7   +8  *  *  *  *  *  * +F :: SUM
0D00 : F9 91 89 1E 4F 00 89 16   53 00 A3 51 00 89 0E 55 :: 552
0D10 : 00 C3 87 DA EB EB A1 51   00 03 C0 03 C0 03 C0 8B :: 7C0
0D20 : D8 03 C0 03 C0 03 D8 A1   4F 00 D1 E8 D1 E8 D1 E8 :: 954
0D30 : D1 E8 03 D8 03 DF C3 B0   49 E8 76 FF 8A C3 E8 7C :: A40
0D40 : FF 8A C7 E8 77 FF A1 4F   00 25 0F 00 03 C0 03 C0 :: 758
0D50 : 03 C0 03 C0 E8 66 FF C3   A1 4F 00 8B 16 53 00 2B :: 6A5
0D60 : D0 A1 51 00 8B 0E 55 00   2B C8 74 16 78 14 3B CA :: 5BE
0D70 : 74 0A 78 08 C6 06 57 00   00 87 D1 C3 C6 06 57 00 :: 55F
0D80 : 01 C3 F7 D9 3B CA 78 08   C6 06 57 00 03 87 CA C3 :: 753
0D90 : C6 06 57 00 02 C3 B0 4C   E8 17 FF A0 57 00 04 08 :: 5E5
0DA0 : E8 1A FF 8B C2 E8 20 FF   8B C1 03 C0 2B C2 E8 17 :: 950
0DB0 : FF 8B C1 2B C2 03 C0 E8   0E FF 8B C1 03 C0 E8 07 :: 8EE
0DC0 : FF C3 B0 6C E8 EB FE C3   E4 A0 A8 04 74 FA 50 58 :: AB8
0DD0 : 50 58 E4 A0 A8 08 75 FA   A0 01 00 A8 01 74 14 BA :: 6D7
0DE0 : 00 A8 A0 00 00 D0 E8 73   04 81 C2 E8 03 E8 33 00 :: 6C0
0DF0 : A0 01 00 A8 02 74 14 BA   00 B0 A0 00 00 D0 E8 73 :: 608

 ADD : +0  *  *  *  *  *  * +7   +8  *  *  *  *  *  * +F :: SUM
0E00 : 04 81 C2 E8 03 E8 1B 00   A0 01 00 A8 04 74 11 BA :: 5C1
0E10 : 00 B8 A0 00 00 D0 E8 73   04 81 C2 E8 03 E8 03 00 :: 6A0
0E20 : 33 C0 C3 FC 06 8E C2 33   FF B9 40 1F 33 C0 F3 AB :: 8E3
0E30 : 07 C3 FC C4 36 19 00 AC   0A C0 75 01 C3 24 07 A2 :: 655
0E40 : 01 00 BB 4F 00 AC 32 E4   03 C0 89 07 43 43 AC 32 :: 584
0E50 : E4 89 07 43 43 AC 32 E4   03 C0 89 07 43 43 AC 32 :: 673
0E60 : E4 89 07 43 43 E8 D4 FD   EB CD 00 00 00 00 00 00 :: 66B
```

次に，UFO（空飛ぶ円盤）を描くサンプルプログラムを示しますので参考にして下さい。

なお，PC-9801F では最初に SCREEN 0, 0 を実行しておいて下さい。

```
1000   ' SAVE"UFO.PAC"
1010 CLS
1020 CLEAR ,&H1D00:DEF SEG=&H1D00:BLOAD "TEC3D.OBJ"
1030 GOSUB *INITIALIZE
1040 ' SET UFO 3D DATA
1050 DIM SN(24),CN(24)
1060  PRINT "JUST WAIT A MOMENT. NOW WRITING UFO DATA."
1070  FOR I=0 TO 24
1080   SN(I)=SIN(2*3.14159/24*I)
1090   CN(I)=COS(2*3.14159/24*I)
1100  NEXT
1110  IB=&H1000 : ' INPUT BUFFER POINTER TOP
1120  RESTORE *UFO.DATA
1130 *SET.UFO
1140  READ R,Y,DY : ' RADIUS AXIS_Y LINE_COLOR
1150  IF R=255 THEN *UFO.PART2
1160  START=&H50:'START POINT CODE
1170  FOR I=0 TO 24:CL=CL+1
1180   POKE IB,START+(CL MOD 6+1):IB=IB+1
1190   Q=R*CN(I):GOSUB *WORD:POKE IB,H:POKE IB+1,L:IB=IB+2
1200   Q=Y      :GOSUB *WORD:POKE IB,H:POKE IB+1,L:IB=IB+2
1210   Q=R*SN(I):GOSUB *WORD:POKE IB,H:POKE IB+1,L:IB=IB+2
1220   START=&H60:'CONNECT POINT CODE
1230  NEXT
1240 GOTO *SET.UFO
1250 *UFO.DATA:DATA 100,32,7   ,92,0,7   ,35,-20,5
1260           DATA 35,-40,2  ,28,-48,2 ,255,255,255
1270 '
1280 *UFO.PART2
1290 '
1300  FOR J=0 TO 24
1310  RESTORE *UFO.DATA:START=&H50:CL=CL+1
1320 *UFO.PART2.LOOP
1330  READ R,Y,DY:IF R=255 THEN *UFO.NEXT
1340  POKE IB,START+(CL MOD 6+1):IB=IB+1
1350   Q=R*SN(J):GOSUB *WORD:POKE IB,H:POKE IB+1,L:IB=IB+2
1360   Q=Y      :GOSUB *WORD:POKE IB,H:POKE IB+1,L:IB=IB+2
1370   Q=R*CN(J):GOSUB *WORD:POKE IB,H:POKE IB+1,L:IB=IB+2
1380  START=&H60
1390  GOTO *UFO.PART2.LOOP
1400 *UFO.NEXT
1410  NEXT
1420  POKE IB,255:IB=IB+1
1430 '
1440  FOR I=355 TO 0 STEP -5
1450   Z1=I:B0=I-180:GOSUB *CONVERT
1460   FOR K=1 TO 3
1470    CL=CCL
1480    FOR J=1 TO 7:CL=CL+1
1490     IF CL=8 THEN CL=1
1500     COLOR=(J,CL)
1510    NEXT:CCL=CCL+1:IF CCL=8 THEN CCL=1
1520   NEXT
1530  NEXT
1540 END
1550 ' SAVE"CTRL3D.PAC",A
1560 '
1570 ' Parameter
1580 '
1590 *WORD
1600  IF Q<0 THEN Q=65535!+Q
```

```
1610  H=INT(Q/256) : L=Q-H*256
1620  RETURN
1630 '
1640 *ANGLE
1650  Q=INT(Q/180*128)
1660  IF Q<0 THEN Q=256+Q
1670  RETURN
1680 '
1690 *CONVERT
1700  Q=X0 : GOSUB *WORD : POKE  6,L:POKE  8,H: ' X
1710  Q=X1 : GOSUB *WORD : POKE 12,L:POKE 13,H: ' X1
1720  Q=Y0 : GOSUB *WORD : POKE  9,L:POKE 10,H: ' Y
1730  Q=Y1 : GOSUB *WORD : POKE 14,L:POKE 15,H: ' Y1
1740  Q=Z0 : GOSUB *WORD : POKE 11,L:POKE 12,H: ' Z
1750  Q=Z1 : GOSUB *WORD : POKE 16,L:POKE 17,H: ' Z1
1760  Q=P0 : GOSUB *ANGLE: POKE 18,Q : ' PITCH
1770  Q=B0 : GOSUB *ANGLE: POKE 19,Q : ' BANK
1780  Q=H0 : GOSUB *ANGLE: POKE 20,Q : ' HEADING
1790 IF SCRN=0 THEN POKE 0,2:SCREEN ,,,2:SCRN=1
           ELSE POKE 0,1:SCREEN ,,,1:SCRN=0
1800  POKE 1,7:A=USR2(0) : A=USR(0) : A=USR1(0)
1810 RETURN
1820 '
1830 ' Transfer data into input buffer
1840 '
1850 *TR.DATA
1860  IB=&H1000 : ' INPUT BUFFER LOCATION
1870 *TR.DATA.LOOP
1880  READ C0
1890  POKE IB,C0:IB=IB+1
1900  IF C0=255 THEN RETURN
1910   FOR I=1 TO 3
1920    READ Q : GOSUB *WORD
1930    POKE IB,H:IB=IB+1
1940    POKE IB,L:IB=IB+1
1950   NEXT I
1960  GOTO *TR.DATA.LOOP
1970 '
1980 ' Initialize 3D-2D converter
1990 '
2000 *INITIALIZE
2010 '
2020  DEF USR0=&H80 :REM 3D->2D
2030  DEF USR1=&H83 :REM DISPLAY CRT
2040  DEF USR2=&H86 :REM CLEAR CRT
2050 '
2060  POKE &H15,0 : ' SET IBP (Input Buffer Pointer)=&H1000
2070  POKE &H16,&H10
2080  POKE &H17,0 : POKE &H18,&H1D : ' SEGMENT ADDRESS=&H1D00
2090  POKE &H19,0 : ' SET OBP (Output Buffer Pointer)=&H2000
2100  POKE &H1A,&H20
2110  POKE &H1B,0 : POKE &H1C,&H1D : ' SEGMENT ADDRESS=&H1D00
2120 RETURN
```

第５章 キー入力

5-1 キー入力バッファ

5-1-1 BIOSキー入力バッファ
5-1-2 インタプリタのキー入力バッファ

5-2 ファンクションキー

5-2-1 ファンクションキーの構造
5-2-2 ファンクションキーの初期化
5-2-3 ファンクションキーの退避・復活

5-3 キースキャン方式

5-4 キー入力のセンス

5-5 キー入力方法・キーセンス比較表

第5章　キー入力

5-1 キー入力バッファ

5-1-1 BIOS のキー入力バッファ

キー入力バッファはキュー入力形式をとっています。1文字を2バイトとして記憶し，キューの長さは32バイトです。

キュー関係はセグメント 0000H（システム共通域）にあります。

・キューアドレス：502H～521H
・キューバッファ・ポインタ：524H，525H
　（ヘッド・ポインタ　HEAD　POINTER）
・キューバッファ最終アドレス＋1：526H，527H
　（テイル・ポインタ　TAIL　POINTER）
・キューバッファ文字数　：528H
　（バッファ・カウンタ）
・キーボード入力時におけるエラーリトライ数：529H

次にこの概念図（図5-1）を示しておきます。

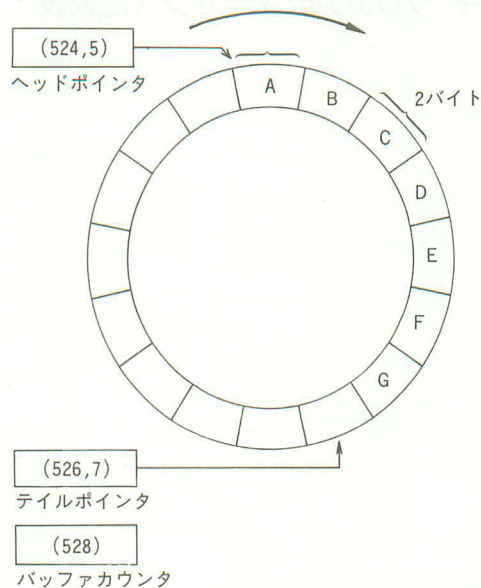



図5-1 キー入力バッファ概念図

個々の文字コードは，２バイトで成っています。下位１バイトは，マニュアルに記載されているキャラクタ・コード表のものと同じですが，上位１バイトは，図5-2のように，キーボードの左上すみの ESC キーから順番につけられた，キーコードとなっています。

STOP 60 | COPY 61 | f・1 62 | f・2 63 | f・3 64 | f・4 65 | f・5 66 | f・6 67 | f・7 68 | f・8 69 | f・9 6A | f・10 6B | ROLL UP 36 | ROLL DOWN 37

ESC 00 | 1 01 | 2 02 | 3 03 | 4 04 | 5 05 | 6 06 | 7 07 | 8 08 | 9 09 | 0 0A | − 0B | ^ 0C | ¥ 0D | BS 0E | INS 38 | DEL 39

TAB 0F | Q 10 | W 11 | E 12 | R 13 | T 14 | Y 15 | U 16 | I 17 | O 18 | P 19 | @ 1A | [1B | ↵ 1C

CTRL 74 | CAPS 71 | A 1D | S 1E | D 1F | F 20 | G 21 | H 22 | J 23 | K 24 | L 25 | ; 26 | : 27 |] 28 | ↑ 3A

SHIFT 70 | Z 29 | X 2A | C 2B | V 2C | B 2D | N 2E | M 2F | < 30 | > 31 | ? 32 | _ 33 | SHIFT 70 | ← 3B | → 3C

カナ 72 | GRAPH 73 | 34 | XFER 35 | ↓ 3D

キーコード

HOME CLR 3E	HELP 3F	− 40	／ 41
7 42	8 43	9 44	＊ 45
4 46	5 47	6 48	＋ 49
1 4A	2 4B	3 4C	＝ 4D
0 4E	, 4F	・ 50	↵ 1C

テンキー

図5-2 キーボードとキーコード

実際に，キューをのぞいてみましょう。

```
MON
hJC0 ←――システム共通域のセグメントにする
hJD502,529
0502 44 1F 35 47 30 4E 32 4B 2C 4F 35 47 32 4B 39 44     D 5G0N2K,O5G2K9D
0512 0D 1C 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0522 26 0B 14 05 14 05 00 00 ←キー入力エラーリトライ数           &
           └ヘッド └テイル └バッファ
            ポインタ ポインタ ポインタ
```

バッファの先頭から，2バイトおきにみると，

D 502, 529 CR

という文字がみえますね。

ヘッドポインタ＝テイルポインタ＝514 H, バッファカウンタ(528)＝0 ですから，これらの文字はすでに取り出されたことになります。このダンプをみるために

D 502, 529 CR

としたときの残がいですね。

各1文字をくわしくみると，最初のDには，1Fというコードがくっついていますね。これは，先程のキーコード表の番号です。大文字のDのキーの部分をみると，

ほら，1Fですね。残りの文字も対応づけてみて下さい。

5-1-2 インタプリタのキー入力バッファ

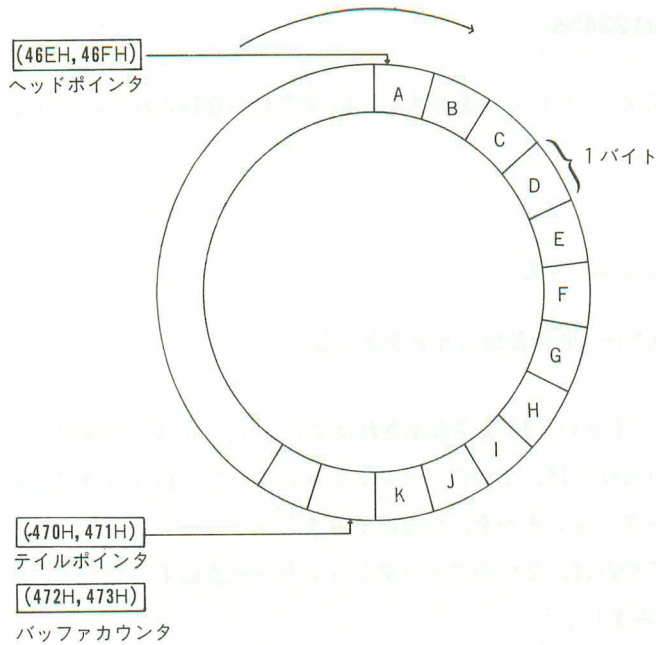
BIOSのキー入力バッファは16文字分のバッファしかないため16文字しか先行入力できません。ところが，N88-BASIC (86) では，32文字の先行入力ができます。これは，インタプリタ用に，もう1つキー入力バッファがあるためです。

インタプリタ用のキー入力バッファは，テキストセグメント（セグメント60 H)内にあります。

・キューアドレス：4 E 0 H～4 FFH

・キューバッファ・ポインタ：46 EH，46 FH(ヘッド ポインタ)

・キューバッファ最終アドレス＋１：470 H，471 H(テイル・ポインタ)
・キューバッファ文字数：472 H，473 H(バッファ・カウンタ)

次にこの概念図を示しておきます。

今度は，１文字１バイトで，そのアスキーコードが入ります。

5-2 ファンクションキー

5-2-1 ファンクションキーの構造

ファンクションキーの内容は，テキストセグメント（60H）の 379 H～42A Hに格納されています。

f-1	379H～389H	f-6	3D3H～3E3H
f-2	38BH～39BH	f-7	3E5H～3F5H
f-3	39DH～3ADH	f-8	3F7H～407H
f-4	3AFH～3BFH	f-9	409H～418H
f-5	3C1H～3D1H	f-10	41AH～42AH

各キーにつき，17バイトが割り当てられていますが，各バッファの先頭の1バイトで，キーに格納されている文字数を示し，終わりの1バイトが必ず00（文字の出力は先頭の文字数でやっているので，この00は，あまり意味がありませんが，）となっているので，定義できるのは15文字までとなっています。

各キーバッファは，スペース（コード20H）で区切られています。（あとで説明しますが，このスペースがくせものです。）

```
KEY 1,"1234567890123456"
Ok
```

で，16文字，入れましたが，f･1 キーをおすと，15文字しか登録されていません。

ところが，

```
DEF SEG=&H60
Ok
POKE &H379,16←16文字にする
Ok
POKE &H389,ASC("6")←00の部分に6をかきこむ
Ok
```

ここで f･1 キーをおして下さい。16文字表示されたでしょう。00は一見区切り記号のようですが，意味はないのです。これも使えば，1つのファンクションキーで，16文字まで登録できます。

今度は，2つのファンクションキーを，つなげてみましょう。

先頭の文字数をかえてやれば，2つのファンクションキーを連結することができます。

f･1とf･2をつなげてみましょう。

```
KEY 1,"ABCDEFGHIJKLMNO"
Ok
KEY 2,"123456789012345"
Ok
```

この状態では，f･1 を押してもアルファベットしか出ません。ここで，次のようにします。

```
DEF SEG=&H60
Ok
POKE &H379,&H22←34文字にする。
Ok
POKE &H389,ASC("@")←00のところに@をかく。
Ok
POKE &H38B,ASC("+")←f･2の文字数のところに+をかく。
Ok
```

ここで f･1 をおします。

```
ABCDEFGHIJKLMNO@_+1234567890123■
                ↑                ↑
            区切りマーク      カーソル点滅
```

f·1 と f·2 がつながってしまいました。こうすることによって17文字以上の定義もできます。

しかしよく見ると，@と＋の間に＿(下線)が見えますね。ここは，38AH番地に相当します。モニタでみてみましょう。

```
MON
hJC60
hJD38A
038A 20 ・・・
```

コード20Hは空白です。下線のコードは，80Hですからおかしいですね。この20を41にかえてみましょう。

```
hJS38A
038A 20-41
hJD38A
038A 41 ・・・
```

確かに41になっています。これで下線は，41H（A）になるかな？

```
hJ^B
Ok
```

f·1 キーをおしてみましょう。

```
ABCDEFGHIJKLMNO_+1234567890123
```

やっぱり，下線が表示されますね。38AH番地はどうなったでしょうか？

```
MON
hJC60
hJD38A
038A 20 ・・・
hJ
```

あれっ。いつのまにか20 H（空白）に戻っています。

数字の部分も0123で終わっています。本当は，012345まであったのですが，

ファンクションキーをつなげても，31文字までしか登録できないからです。

しかし，この下線の区り記号なんとか消せないものでしょうか。

```
MON
hJC60
hJS5C7
05C7 80-00
```

さて今度はどうですか，f･1 をおして下さい。

```
ABCDEFGHIJKLMNO@+1234567890123
```

消えた。区切りの下線が消えましたね。

5 C 7番地に00を書いただけですね。

テキストセグメント60 HのBASICのワークエリア中の5 C 6 H～5 CFH番地は，ファンクションキーの状態をあらわすフラグだったのです。

その意味は，

- 80 H…通常のファンクションキー
- 20 H…キー割り込みON
- 00 H…ファンクションキーとしての働きを殺す。

ということだったのです。各アドレスとキーは次のように対応しています。

5C6H………f-1	5CBH………f-6
5C7H………f-2	5CCH………f-7
5C8H………f-3	5CDH………f-8
5C9H………f-4	5CEH………f-9
5CAH………f-5	5CFH………f-10

さっきは，f･2 キーの機能を殺したので，区切りの下線が消えたのですね。このとき，モニターで書き直しましたが，KEY　（2）　ON

としても下線を消すことができます。ただし，下線のかわりに空白を表示します。

```
ABCDEFGHIJKLMNO@ +1234567890123
```

空白になっている

なぜモニターで書き直したかというと，POKEを使っても書き直せなかったからなのです。5C7Hに00を書き込めば，区切りのスペースは無視されるのですが，どうすればよいのでしょうか？

```
POKE &H5C7,0
```

では，5C7H番地に書き込むことはできませんが，

```
POKE &H38A,0
```

とすればよいのです。f·1キーをおして下さい。区切りの下線もスペースも消えましたね。モニタで，5C6H番地をみてみましょう。

```
MON
h]D5C6
05C6 80 00 80 80 80 80 80 80 80 80
h]
```

ほら，00になっていますよ，38AH番地に00を書き込んだはずなのに変ですね。これは，ファンクションキーの状態フラグは，5C6H〜5CFHにありますが，その書き換えは，各ファンクションキーバッファの前にある，20H（スペース）の書き込んである番地を使って行うのです。

ファンクションキーの状態フラグは，次の番地にPOKEするとOKです。

f-1…………378H　　f-6…………3D2H
f-2…………38AH　　f-7…………3E4H
f-3…………39CH　　f-8…………3F6H
f-4…………3AEH　　f-9…………408H
f-5…………3C0H　　f-10…………419H

しかし，実際のフラグはf-1〜f-10それぞれ5C6H〜5CFHにあります。

5-2-2 ファンクションキーの初期化

ファンクションキーの初期データは，ROMの中に入っています。しかし，ROMのバージョンによって，その位置は，様々に違います。しかし，ROM内ルーチンCALLのインタラプトのベクタを用いると，ROMのバージョンによらずに，初期データの位置を知ることができます。アドレスが分かれば，あとは，そのデータをRAMのバッファに転送してやるだけでよいのです。

これをBASICで行うと次のようになります。

ファンクションキー初期化プログラム

```
0 'SAVE"FNKEY.INI"
100 *F.INIT
110 '
120 ' FUNCTION KEY INITIALIZE
130 '     INDEPENDENT ROM VERSION
140 '
150 DIM S(179)
```

```
160 '
170 ' GET THE ADDRESS OF FUNCTION KEYS' DATA
180 '
190 DEF SEG=0
200  O1=PEEK(&H310)+PEEK(&H311)*256
210  S1=PEEK(&H312)+PEEK(&H313)*256
220 DEF SEG=S1
230  O2=PEEK(O1+7)+PEEK(O1+8)*256:O3=O2+42
240  O4=PEEK(O3)+PEEK(O3+1)*256
250  '
252  FL=-1:N=0:WHILE FL
254   IF PEEK(O4)=&H80 AND PEEK(O4+1)=6 AND PEEK(O4+2)=ASC("l")
      AND PEEK(O4+3)=ASC("o") AND PEEK(O4+4)=ASC("a") THEN FL=0
256  O4=O4+1  :WEND
258  O5=O4-1
260 '
270 '  GET THE FUNCTION KEYS' DATA INTO S(I)
280 '
290 FOR I=0 TO 179
300  S(I)=PEEK(O5+I)
310 NEXT
320 '
330 '  SET THE DATA TO RAM BUFFER
340 '
350 DEF SEG=&H60
360  FOR I=0 TO 179
370   POKE &H378+I,S(I)
380  NEXT
390 '
400 '  INITIALIZE THE SCREEN KEY LIST
410 '
420 CLS
430 RETURN
```

190行から250行がファンクションキーの初期データのTOPアドレスを求める部分です。INT C4Hルーチンのベクタ（セグメント0の310H～313Hにあります）を用いてROM内のそのルーチンのエントリからの偏差を計算しています。

これはサブルーチンになっていて，必要な所にGOSUB　*F.INITを入れておくとファンクションキーが初期化されます。

次に，これを，マシン語で行った例を示します。

```
0000 31C0          XOR     AX,AX
0002 8ED8          MOV     DS,AX
0004 8B1E1003      MOV     BX,[0310]
0008 8E1E1203      MOV     DS,[0312]
000C 83C307        ADD     BX,0007
000F 8B07          MOV     AX,[BX]
0011 052A00        ADD     AX,002A
0014 89C3          MOV     BX,AX
0016 8B1F          MOV     BX,[BX]
0018 B88006        MOV     AX,0680
001B 3B07          CMP     AX,[BX]
```

```
001D 7403           JE      0022
001F 43             INC     BX
0020 EBF9           JMPS    001B
0022 FC             CLD
0023 16             PUSH    SS
0024 07             POP     ES
0025 B9B400         MOV     CX,00B4
0028 BF7803         MOV     DI,0378
002B 89DE           MOV     SI,BX
002D F3             REP
002E A4             MOVSB
002F 31C0           XOR     AX,AX
0031 CF             IRET
```

マシン語格納領域を,

```
CLEAR ,&H1D00
Ok
```

で確保し, Disk BASIC ならばモニタにアセンブラがありますから,

```
hJA0
```

で入力して下さい。ROM 版の人は, アドレスのすぐ右のオブジェクト・コードをモニタのSコマンドを使って,

```
hJS0
0000 00-31 00-C0 00-
```

と入力して下さい。

実行は, モニタのまま,

```
hJG0,28
```

とするか,

```
DEF USR=0
Ok

A=USR(0)
Ok
```

又は,

```
F.INIT=0
Ok

CALL F.INIT
Ok
```

として下さい。

5-2-3 ファンクションキーの退避・復活

ビジネスプログラムなどで, ファンクションキーの内容をひんぱんに書き換えるものがありますが, よく使うキーの内容は, あるメモリ領域に退避して保存しておき, 必要なときに復活すると便利です。

そこで, CALL 文を用いて, そのパラメータが0のときは退避, 1のときは復活させるルーチンを紹介します。使い方は次のとおりです。

```
DEF SEG=&H1F00
FK=0
A%=0(退避)　またはA%=1(復活)
CALL FK(A%)
```

次にそのサンプルプログラムを示します。

ファンクションキー退避・復活プログラム

```
1 ' save "FK.bas"
100 CLEAR ,&H1F00 :WIDTH 80,25
110 DEF SEG =&H1F00 : FK=0
120 FOR AD=0 TO &H22
130  READ D$ : D=VAL("&H"+D$)
140  POKE AD,D
150 NEXT AD
160 A%=0 : CALL FK(A%) ' Save
170 FOR I=1 TO 10 : KEY I,"" :NEXT I
180 FOR I=1 TO 2000 : NEXT I
190 A%=1 : CALL FK(A%) : WIDTH 80 ' Restore
200 END
210 DATA C5,37,8B,04,16,5B,0E,59,0E,1F,BE,78,03,BF,23,00:'0000H
220 DATA 0B,C0,74,04,87,D9,87,F7,8E,DB,8E,C1,FC,B9,5A,00:'0010H
230 DATA F3,A5,CF,90,90,90,90,90,90,90,90,90,90,90,90,90:'0020H
```

※しかし表示は変りませんのでシフトキーを押すか，CONSOLE,, 1を実行して下さい。

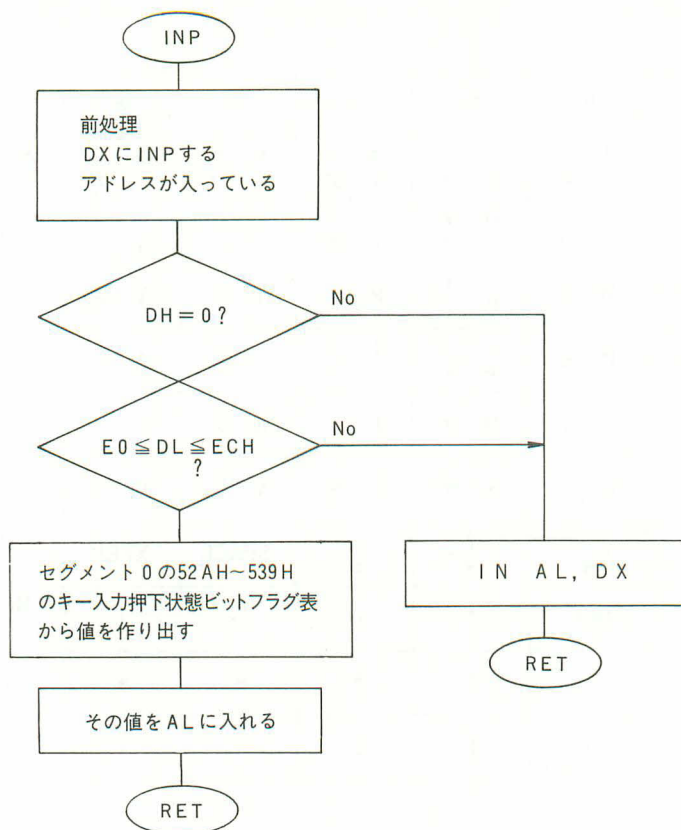
5-3 キー・スキャン方式

PC-9801のユーザーズ・マニュアルを見ると，「19.4.1　スキャン方式」という章があり，I/Oポートとキーの対応表がのっています。

確かに，BASICのINP関数を用いて行うと，このとうり，対応するビットが0になりますが，マシン語のIN命令で直接行うと，うまくいきません。

一体どうなっているのかというと，I/OポートE0H～ECHは実際には，使われていないのです。INP関数の処理ルーチンは，I/Oアドレスを抽出した時点で，E0H～ECH以外は，

```
EC                IN  AL,DX
```

を行い，実際のI/Oアドレスポートから直接値を取り込みますが，E0H～ECHの場合は，セグメント0(システム共通域)のオフセット52AH番地～539H番地の16バイトの入力キーに対応したビット表を用いて，その値を決定しているのです。次にINP関数のフローチャートを示しておきます。

BASICのINP関数のフローチャート

ですから，E0H～ECHは使っていないのです。

マシン語でキースキャンをするときも，I/Oポートのにせポート E0H～ECH を使ってはいけません。

セグメント 0000 H のアドレス 52AH～539H 番地の表を使わなければなりません。次にその対応表を示します。

セグメント0

キーコードグループ	ビット アドレス	b0	b1	b2	b3	b4	b5	b6	b7
0	52A	ESC	! 1 ヌ	" 2 フ	# ァ 3 ア	$ ゥ 4 ウ	% ェ 5 エ	& ォ 6 オ	, ャ 7 ヤ
1	52B	(ュ 8 ユ	) ョ 9 ヨ	0 ヲ ワ	= - ホ	∧ へ	¦ ¥ ー	BS	TAB
2	52C	Q タ	W テ	ィ E イ	R ス	T カ	Y ン	U ナ	I ニ
3	52D	O ラ	P セ	~ @ 〃	{ 「 [°	↵	A チ	S ト	D シ
4	52E	F ハ	G キ	H ク	J マ	K ノ	L リ	+ ; レ	* : ケ
5	52F	¦ 」] ム	Z ッ ツ	X サ	C ソ	V ヒ	B コ	N ミ	M モ
6	530	< , ネ ,	> 。 . ル	? / メ ・	― ロ	SPACE	XFER	ROLL UP	ROLL DOWN
7	531	INS	DEL	↑	←	→	↓	HOME CLR	HELP
8	532	-	/	7	8	9	*	4	5
9	533	6	+	1	2	3	=	0	,
A	534	.							
B	535								
C	536	STOP	COPY	f・1	f・2	f・3	f・4	f・5	f・6
D	537	f・7	f・8	f・9	f・10				
E	538	SHIFT	CAPS	カナ	GRAPH	CTRL			
F	539								

（8〜A：テンキの部分）

おされているキーのビットが立ちます。↑キーがおされたとすると，531H番地の内容が04になります。↑と←が同時におされたとすると0Cになります。

表5-3 キースキャン対応表

538H番地は，特殊キーですが，特殊キーをまとめたものが53AH番地にもあります。即ち，53AH番地の内容と538H番地の内容は同じです。

このテーブルを更新するには，ROM内ルーチンの「キー・センス」を呼ばなければ，なりません。

キー・センスルーチンの呼び方は，

```
          MOV DI,4BH
          CALL ROM_CALL
                 :
                 :
ROM_CALL: PUSH DS
          PUSH SS
          POP  DS
          INT  0C4H
          POP  DS
          RET
```

とします。

次に，マシン語ルーチンで，STOP キーチェックを行うプログラムを紹介します。

これは，PC-8001 の ROM 内ルーチン

0CF1H　　ストップキーチェック

に相当するもので，

このルーチンの出力は

CF＝1………STOP キーが押されている
CF＝0………STOP キーが押されていない

です。

ストップキーチェックルーチン

```
0000 06         PUSH    ES
0001 50         PUSH    AX
0002 16         PUSH    SS
0003 1F         POP     DS          ; DS=60 H
0004 31C0       XOR     AX,AX       ; ES=0 テキストセグメント
0006 8EC0       MOV     ES,AX
0008 BF4B00     MOV     DI,004B     ; キーセンス ROM 内ルーチンコール
000B 06         PUSH    ES
000C CDC4       INT     C4
000E 07         POP     ES
000F 26         ES:
0010 A13605     MOV     AX,[0536]; ストップキーのビットが 1 か？
0013 250100     AND     AX,0001
0016 F9         STC                 ; CF=1
0017 7501       JNE     001A
0019 F5         CMC                 ; CF=0
001A 58         POP     AX
001B 07         POP     ES
001C C3         RET
```

このプログラムは，マシン語ルーチンの中で，STOP キーの押下を調べるものですから，BASIC から CALL しても全く意味がありません。

100 H 番地からチェック用のルーチンを作って確認してみましょう。

```
0100 E8FDFE          CALL 0000; ストップキーチェックルーチンを呼ぶ
0103 73FB            JAE 0100
0105 CF              IRET      ; ストップキーが押されるまでLOOPする
```

h] G 100,105

で実行できます。STOP キーを押すと再びモニタのプロンプト

h]

が出てきたでしょう。

STOP キーチェックルーチンを使って，自分の作ったマシン語ルーチンで，STOP キーを押すと停止するとか，ESC キーを押すと一時停止するなど応用して下さい。

5-4 キー入力のセンス

前節で「キー・センス」について少し触れましたが，もっと簡単なインタラプトコールの方法がありますので紹介します。キーボードの上のキーはすべて前節の表5-3の左側に00H～0FHまでのキーコードグループのいずれかに属しています。ここで述べるのは，該当するキーコードグループのキー入力状態を調べ，押されているキーは対応ビットを1とし，押されていないキーは0として，その状態を通知するものです。

このキー入力センスの呼び方は次のとおりです。

```
MOV AH,04H
MOV AL, キーコードグループ番号(00H-0FH)
INT 18H
```

結果はAHにキーコードグループ内の8つのキーの押下状態を8ビットで格納されてきます。

ここで前節のストップキーチェックと同様のことをプログラミングしてみましょう。ストップキーはコードグループ0CHにありますので次のようになります。

```
      PUSH SS
      POP  DS          ;DS<=SS
      MOV  AH,04H
      MOV  AL,0CH
      INT  18H
      AND  AH,01H
      STC
      JNE  BACK
      CMC
BACK: RET
```

5-5 キー入力方法・キーセンス比較表

BASICでキー入力のプログラミングをする際の参考として，入力やセンスの方法を一覧表としてとめました。

★文字列の入力方法の比較表

		INPUT	INPUT WAIT	LINE INPUT	LINE INPUT WAIT	INPUT$(X)	INKEY$
		文	文	文	文	関数	関数
入力表示	プロンプト文	可能	可能	可能	可能	不可能	不可能
	プロンプトマーク？	可能	可能	無	無	無	無
	カーソル表示	有	有	有	有	有	可能＊1
	エコーバック	有	有	有	有	無	無
	入力待ち	待つ	指定した時間だけ待つ	待つ	指定した時間だけ待つ	待つ	待たない
データ入力	，の入力	ダブルクォートで囲めば可	ダブルクォートで囲めば可	可能	可能	可能	可能
	〃の入力	文字列の最初でなければ可	文字列の最初でなければ可	可能	可能	可能	可能
	コントロールコードカーソルキーの入力	不可	不可	不可	不可	可能	可能
	複数の変数への入力	可能	可能	不可	不可	不可	不可
	入力文字数	254文字以内	254文字以内	254文字以内	254文字以内	指定した文字数(255文字以内)	1 文字
	入力終了	↵キー	↵キー	↵キー	↵キー	自動	——
	入力文字なしで↵キー	ヌルストリング	ヌルストリング	ヌルストリング	ヌルストリング	CHR$(13)	CHR$(13)
STOPキー	BREAK表示	あり	あり	あり	あり	なし	あり＊2
	CONTによる再開	可能	可能	可能	可能	不可	可能＊2

＊1 ：通常はカーソルは表示されませんが、表示させることも可能です。第14章ランダムテクニックを見て下さい。

＊2 ：INKEY$がブレークされたりCONTされたりするわけではありません。

INKEY$自体はSTOPキーをCHR$(3)として受け付けることも可能です。

★キーセンス比較表

	INPUT$(1)	INKEY$	INP(X)	WAIT
入力待ち	待つ	待たない	待たない	待つ
STOPキー	中断(CONT不可)	中断*	中断*	中断しない
複数キーの同時入力	不可	不可	可能	可能な場合もある
入力文字の判断	簡単	簡単	複雑	複雑
カーソル表示	表示する	表示させることも可	表示させることも可	表示させることも可
キー割り込み	きかない	きく*	きく*	きかない

＊入力待ちがないためINKEY$, INP(X)自体はSTOPキーやキー割り込みとは関係ありません。

第 6 章　カセットファイル

6-1　CMTインターフェイス

6-2　CMTとのデータ転送の仕様

6-3　データフォーマット

　6-3-1　プログラムファイル
　6-3-2　データファイル
　6-3-3　マシン語ファイル

6-4　内部ルーチン

　6-4-1　データ書き込み
　6-4-2　データ読み込み

6-5　マシン語によるセーブ・ロード

6-6　BASICとマシン語を一度にSAVE・LOAD

6-7　データの高速セーブ・ロード

第6章　カセットファイル

6-1　CMTインターフェイス

PC-9801には，CMT インターフェースボード（PC-9801-03）があり，オーディオカセットへのデータ書き込みおよびカセットからのデータ読み込みが行えます。このインターフェイスの入出力信号は，CMT インターフェイスボードのカセット用コネクタ（8ピンのDIN コネクタ）に接続されています。カセットレコーダとCMT インターフェイスとを接続するには，カセットケーブル(添付のケーブルまたはPC-8093）を用います。

6-2　CMTとのデータ転送の仕様

CMT とのデータ転送は，次のような仕様で行われます。

通信速度：600ボー／1200ボー
　　　　　（ソフトで選択可能）
通信方式：調歩式（アシンクロナスモード），全二重
ストップビット：2ビット
符号単位数：8ビット

6-3　データフォーマット

6-3-1 プログラムファイル

N_{88}-BASIC (86) のプログラムをカセットにSAVE したときのフォーマットは次のようになります。

モータオン ウェイト2.5秒		D3×10回				エンドマーク	モータオフ
WAIT スペース 0.6秒	WAIT マーク 1.9秒	ヘッダー D3が 10バイト	ファイル名 6バイト	WAIT マーク 0.1秒	プログラム 本体 最後の3バイトは00	00×9	WAIT 1.3秒

まずテープレコーダのモータをオンにし，テープ走行が安定するまで待った後，D 3 H を 10 個書き込みます。次に，ファイル名を書き込みますが，6文字に未たないときは，その後に 00 H をつけ加え6バイトにします。それから再びウエイトがあります。これは，LOAD 時に〝Found〃や〝Skip〃

などの表示をするための時間です。

さて次がいよいよプログラム本体です。本体が書き込まれた後の最後の3バイトは00Hになっていて，次に続く9つの00Hとあわせ，12個の00Hでエンドマークとなります。

6-3-2 データファイル

モータオン				モータオフ
WAIT スペース 0.6秒	WAIT マーク 1.9秒	ヘッダー "9CH" 6バイト	データ 本体	WAIT マーク 1.3秒

テープ走行安定のためのウエイトが2.5秒あるのは，プログラムファイルと同じで，9CHを6個書き込んでヘッダーを作り，データファイルであることを示します。

続いて，データ本体が書き込まれますが，数値の場合もすべて文字列に直して書き込まれます。

6-3-3 マシン語ファイル

モータオン	ヘッドブロック						データブロック1				データブロック2	エンドブロック				モータオフ
WAIT マーク 3.8秒	"3A" 1	AH 1	AL 1	CS 1	"3A" 1	N 1	データ 1-255 バイト	CS 1	"3A"	N 1	1-255 バイト	CS 1	"3A" 1	N "00" 1	CS "00" 1	WAIT マーク 1.3秒

まずウエイトがあった後，ヘッドブロックがあります。これは，次のようになっています。

3AHを1個書き込む

AH =スタートアドレス（High）

AL =スタートアドレス（Low）

CS =チェックサム

N =データ数（00～FFH）

6-4 内部ルーチン

ここではアセンブリ言語（マシン語レベル）でカセットファイルへの書き込み，読み込みを行うプログラムを紹介します。

6-4-1 データ書き込み

CMT のモータをオンにし，データの書き込みを可能にし，データを書き込みます。

```
MOV AH,03H                          ┐モータ・オン
MOV AL,00H (600ボー)                │書き込み可能
       80H (1200ボー)               │
INT 1AH                             ┘
```

```
MOV AH,04H                          ┐
MOV AL,41H (出力データ1バイト       │書き込み
            例では“A”)              │
INT 1AH                             ┘
```

6-4-2 データ読み込み

CMT のモータをオンにして,読み取り可能に設定し，データを読み取ります。

```
MOV AH,02H                          ┐モータ・オン
MOV AL,00H (600ボー)                │読み取り可能
       80H (1200ボー)               │
INT 1AH                             ┘
```

```
MOV AH,05H                          ┐
MOV AL,00H (リードエラー通知)       │読み取り
       01H (エラーを無視)           │
INT 1AH                             ┘
```

```
AH=00H (正常終了)                   ┐
   02H (タイムアウト)               │結果
   27H (テープリードエラー)         │
AL=(入力データ)                     ┘
```

6-4-3 CMT のモータオフ

CMT のモータをオフにします。

```
MOV AH,01H  ┐モータ・オフ
INT 1AH     ┘終了
```

6-5 マシン語によるセーブ・ロード

内部ルーチンの使い方が分かったところで実際のプログラミングに挑戦してみましょう。アスキーコードの0から255までをセグメント1F00H，オフセット100 Hから書き込んでおき，まずそれをセーブします。次にそのデータを同じセグメントのオフセット 200 H からロードしてみます。

使い方は，それぞれ

```
DEF SEG=&H1F00
A=0:CALL A
```

で OK です。

カセットへの書き込み

```
                              ;===================
                              ; CASSETTE WRITE
                              ; SAMPLE
                              ;===================
                              ;
0000 32C0                     DATA:   XOR AL,AL          ; AL=0
0002 B90001                           MOV CX,100H        ; NO.OF DATA=256
0005 0E                               PUSH CS
0006 07                               POP ES             ; ES=CS
0007 BB0001                           MOV BX,100H        ; OFFSET
000A 268807                   STORE:  MOV ES:[BX],AL     ; AL=0 -- 255
000D FEC0                             INC AL
000F 43                               INC BX
0010 E2F8           000A              LOOP STORE         ; LOOP UNTIL CX=0
                              ;
0012 B403                     CMTON:  MOV AH,03H         ; MOTOR ON AND WRITE READY
0014 B080                             MOV AL,80H         ; 1200 bps
0016 CD1A                             INT 1AH            ; CALL
                              ;
0018 BB0001                   WRITE:  MOV BX,100H
001B B90001                           MOV CX,100H
001E 268A07                   LP:     MOV AL,ES:[BX]     ; AL=DATA
0021 B404                             MOV AH,04H         ; WRITE DATA
0023 CD1A                             INT 1AH            ; CALL
0025 43                               INC BX
0026 E2F6           001E              LOOP LP
0028 CF                               IRET               ; BACK TO BASIC
```

カセットから読み込み

```
                              ;===================
                              ; CASSETTE READ
                              ; SAMPLE
                              ;===================
                              ;
0000 0E                       START:  PUSH CS
0001 07                               POP ES             ; ES=CS
0002 BB0002                           MOV BX,200H        ; DATA STORE OFFSET
0005 B90001                           MOV CX,100H        ; CX=NO. OF DATA
0008 B402                     CMTON:  MOV AH,02H         ; MOTOR ON AND READ READY
000A B080                             MOV AL,80H         ; 1200 bps
000C CD1A                             INT 1AH            ; CALL
                              ;
000E B405                     READ:   MOV AH,05H         ; READ DATA
0010 B000                             MOV AL,00H         ; ERROR CHECK ON
0012 CD1A                             INT 1AH            ; CALL
0014 80FC00                           CMP AH,00H
0017 75F5           000E              JNE READ           ; IF NOT READ THEN TRY AGAIN
```

```
0019 268807                      MOV ES:[BX],AL   ; AL=DATA
001C 43                          INC BX
001D E2EF         000E           LOOP READ
001F CF                          IRET             ; BACK TO BASIC
```

6-6 BASICとマシン語を一度にSAVE・LOAD

BASICのプログラムとマシン語をリンクさせたソフトがよく見うけられます。特にゲームソフトではその傾向が強いようです。カセットテープを使って，BASICとマシン語がリンクされたプログラムをメモリ上にロードするには通常次の2通りがあります。

1. BASICのDATA文中にマシン語を入れておき，それをREAD文で読み出した後，POKE文により書き込む。この方法ですと，BASICのプログラム1つになるため，SAVE，LOADでOKということになります。しかしBASICのプログラムが少しふくらむことになります。

例)
```
1 ' save "poke"
100 DEF SEG=&H60
110 SUB=&H283E
120 FOR I=0 TO 18
130   READ D$ : POKE SUB+I,VAL("&H"+D$)
140 NEXT
150 DATA B8,FF,06,BB,D0,00,4B,83,FB,00
160 DATA 75,FA,48,3D,00,00,75,F1,CF
170 TIME$="00:00:00"
180   CALL SUB
190 PRINT TIME$
200 END
```

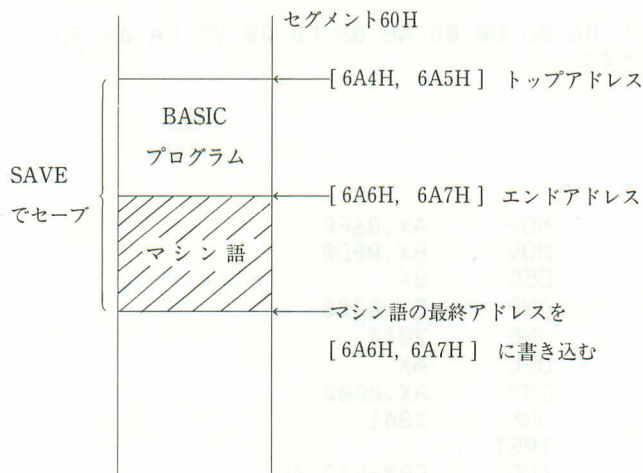
2. MONコマンドによりマシン語モニタに移り，W・Rコマンドでマシン語をセーブ・ロードします。そして，BASICのプログラムをSAVEやLOADします。この方法ですと，BASICのプログラムはふくらみませんがBASICとマシン語を別々にセーブ・ロードしなくてはいけません。

そこで，第3の方法を紹介いたしましょう。これは，1と2の方法の欠点を解決したもので，メモリー上にあるBASICとマシン語のプログラムをSAVEだけで一度にセーブし，またそれをLOADだけでロードできます。

PC-9801では，BASICのプログラムは次のアドレスに格納情報があります。

セグメント=60 H

オフセット=6 A 4 H，6 A 7 H

6A4H, 6A5HにBASICのトップアドレス, 6A6Hと6A7Hにエンドアドレスが格納されています。

考え方としては, BASICのプログラムのすぐ後ろからマシン語を入れておき, その最終アドレスを〔6A6H, 6A7H〕に書き込めばOKです。そうして, SAVEすれば, BASICとマシン語が一度にセーブできるわけです。それでも注意することが3つあります。1つは, BASICの終りには必ず00Hが2つなくてはいけないことです。2つめは, マシン語の中に00Hが10個以上連続していないことです。3つめは, マシン語の最後では00Hが10個以上続いていなければなりません。これは, BASICプログラムのデータフォーマットを参照されたら解かると思います。これは理論的にはそうですが, 実際はそうなっていなくてもいいようです。

それではサンプルプログラムを示しながら実際に試してみることにしましょう。次のBASICプログラムを入力して下さい。なお, ここで示すアドレスはPC-9801のバージョンとシステム構成により異なることがあります。

```
10   BASIC + MACHINE
20 DEF SEG=&H60
30 SUB=&H283E
40 TIME$="00:00:00"
50   CALL SUB
60 PRINT TIME$
70 END
```

```
MON   トップ   エンド
hJC60
hJD6A4
06A4 D8 27 3B 28 FE F3 FE F7 FE F9 00 01 00 01 EE F7     ﾘ´;( 月 秒      ／秒
hJD283B
```

```
283B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
hJD283E←── ここからマシン語を入れる
283E B8 FF 06 BB D0 00 4B 83 FB 00 75 FA 48 3D 00 00     ク  サミ K▬   u H=
hJD284E
284E 75 F1 CF 00 00 00 00 00 00 00 00 00 00 00 00 00     u円マ
hJD285E
285E 00 00 B8 FF 06 BB D0 00 4B 83 FB 00 75 FA 48 3D      ク  サミ K▬   u H=
Ok ↑─── ここをエンドとする
```

ソースリスト

```
283E B8FF06        MOV     AX,06FF
2841 BBD000        MOV     BX,00D0
2844 4B            DEC     BX
2845 83FB00        CMP     BX,0000
2848 75FA          JNE     2844
284A 48            DEC     AX
284B 3D0000        CMP     AX,0000
284E 75F1          JNE     2841
2850 CF            IRET
2851 0000          ADD     [BX+SI],AL
2853 0000          ADD     [BX+SI],AL
2855 0000          ADD     [BX+SI],AL
2857 0000          ADD     [BX+SI],AL
```

マシン語はタイマールーチンで2秒間ウエイトするものです。BASICとマシン語の両方を入力したら今度はエンドポインタを書き替えてセーブしましょう。エンドポインタは，285EHにしてみます。そのまえに本来のエンドポインタ（283BH）を覚えておいて下さい。

ここでSAVEします。がLISTをとってはいけません。暴走するか，リストがメチャクチャになります。

セーブした後，リセットをかけてメモリをクリアしてから本当にロードできるか試してみましょう。なお，LOADした後，またエンドポインタを元にもどさなくてはなりません。なお、第2章のBASICプログラム復活ルーチンを利用してもOKです。

```
MON
hJC60
hJS6A6
06A6 3B-5E 28-28 FE-
hJ^B
Ok
SAVE "CAS:TEST"
Ok
```

```
LOAD "CAS:TEST"
Ok
```

```
MON
h]C60
h]S6A6
06A6 5E-3B 28-28 FE-
h]^B
Ok
```

これまでの方法はちょっと手間がかかります。そこで，6-5のマシン語によるセーブ・ロードを使って1度にBASICとマシン語をセーブ・ロードしてみるのも1つの手です。それは読者の皆様の課題にしておきましょう。なお，以上の方法ではPAINT文を実行するとマシン語部分が破壊されますので要注意（第1章 1-5参照）。

6-7 データの高速セーブ・ロード

カセットファイルにデータを書き出すにはOPEN〝CAS：ファイル名〞FOR OUTPUT AS #1としてファイルをオープンし，PRINT #1，A $のようにします。データが少ないときは1個1個書き出してもそんなに時間はかかりませんが，データが多いときは，けっこう時間をとります。

これはデータを1個書くたびに区切りを入れているためです。次のプログラムを実行してみて下さい。10個のデータを書き込むものです。

```
1 'SAVE "D1.cas"
2 'High Speed SAVE・LOAD
10 DIM A$(10)
20 FOR I=1 TO 10
30 READ A$(I)
40 NEXT I
50 DATA A,B,C,D,E,F,G,H,I,J
60 '
70 TIME$="00:00:00"
80 OPEN "CAS:DEMO1" FOR OUTPUT AS #1
90  FOR I=1 TO 10
100   PRINT #1,A$(I)
110 NEXT I
120 CLOSE #1
130 PRINT TIME$
Ok
run
00:00:29
Ok
```

1200ボーで書いても29秒かかっています。では，そのデータを読んでみましょう。

```
1 'SAVE "D1r.cas"
2 'High Speed SAVE·LOAD
10 '
20 TIME$="00:00:00"
30 OPEN "CAS:DEMO1" FOR INPUT AS #1
40  FOR I=1 TO 10
50   INPUT #1,A$(I):PRINT A$(I)
60 NEXT I
70 CLOSE #1
80 PRINT TIME$
Ok
Ok
run
00:00:31
Ok
```

データリードには31秒かかりました。

ではデータとデータの区切りをつめて，一気に書き込む方法を紹介しましょう。

PRINT＃1，A＄(I)；〝,〟；A＄

(I＋1)；〝,〟・・・というのがポイントでこの方法だと，非常に高速になります。次のサンプルを実行してみて下さい。

```
1 'SAVE "D3.cas"
2 'High Speed SAVE·LOAD
10 DIM A$(10)
20 FOR I=1 TO 10
30 READ A$(I)
40 NEXT I
50 DATA A,B,C,D,E,F,G,H,I,J
60 '
70 OPEN "CAS:DEMO3" FOR OUTPUT AS #1
80 TIME$="00:00:00"
90  FOR I=1 TO 10 STEP 5
100    PRINT #1,A$(I);",";A$(I+1);",";A$(I+2);",";A$(I+3);",";A$(I+4)
110 NEXT I
120 CLOSE #1
130 PRINT TIME$
Ok
run
00:00:06
Ok
```

今度は，たったの6秒になっています。ではそのデータを読んでみましょう。

```
1 'SAVE "D3r.cas"
2 'High Speed SAVE·LOAD
10 '
20 TIME$="00:00:00"
30 OPEN "CAS:DEMO3" FOR INPUT AS #1
```

```
40  FOR I=1 TO 10 STEP 5
50   INPUT #1,A$(I),A$(I+1),A$(I+2),A$(I+3),A$(I+4)
60   PRINT A$(I),A$(I+1),A$(I+2),A$(I+3),A$(I+4)
70 NEXT I
80 CLOSE #1
90 PRINT TIME$
Ok
run
00:00:05
Ok
```

データリードは5秒となっています。

第 7 章　ディスクファイル

7-1　ディスクファイルの構造

7-1-1　ディスクマップ
7-1-2　ディスクアドレスとクラスタとの変換
7-1-3　ディレクトリ
7-1-4　IDセクタ
7-1-5　FAT

7-2　DSKF関数

7-3　標準ディスク

7-3-1　フォーマッティング
7-3-2　ディスクBIOSコマンド

7-4　5インチ・ディスク

7-4-1　概要
7-4-2　ディスクBIOSコマンド

7-5　ディスク・ユーティリティ・プログラム

7-5-1　ファイルインフォメーション
7-5-2　1ファイル転送
7-5-3　ファイルネームソート
7-5-4　オールマイティ・ディスク・ダンプ
7-5-5　簡易ディスクエディター
7-5-6　8インチIDリーダー
7-5-7　インテルHEXファイル・ローダー

第7章　ディスクファイル

7-1　ディスクファイルの構造

7-1-1 ディスクマップ

ここでは，N88-Disk　BASIC(86)におけるディスクのファイル構造についてまとめています。

ディスクの物理構造

ディスク タイプ	サーフェス番号	トラック数	トラック番号	セクタ数	セクタ番号	データ容量	参照図
8インチ両面倍密度	0～1	77	0～76	26	1～26	1.025MB	図1
5インチ両面倍密度	0～1	40	0～39	16	1～16	327.68KB	図2
5インチ両面倍密度倍トラック	0～1	80	0～79	16	1～16	655KB	図3
5インチ片面倍密度	なし	35	0～34	16	1～16	143.36KB	図4
5インチ固定ディスク(5 MB)	0～3	153	0～152	33	1～33	5.17MB	図5
5インチ固定ディスク(10MB)	0～3	310	0～309	33	1～33	10.47MB	図6

ディスクマップ

	サーフェス番号	トラック番号	セクタ番号
IPL	0	0	1～4
Diskコード	0	1～2	すべて
	1	1～2	すべて
ディレクトリ	0	35	1～22
ID	0	35	23
FAT	0	35	24～26

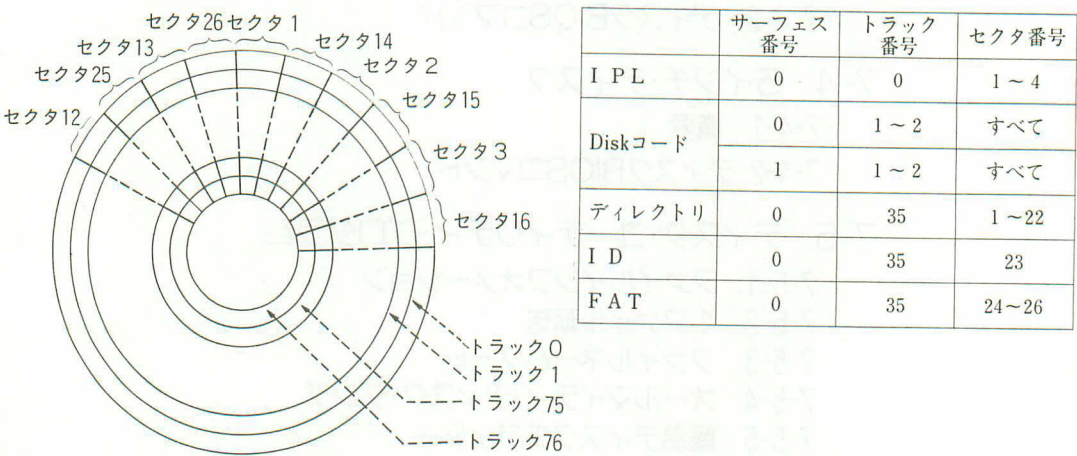



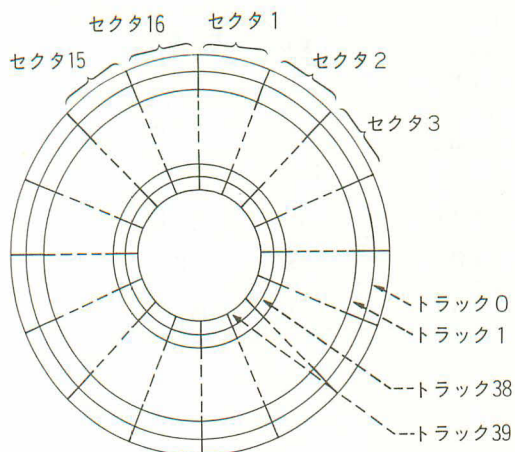
図1　8インチ両面倍密度フロッピィディスク

	サーフェス番号	トラック番号	セクタ番号
IPL	0	0	1～2
Diskコード	0	0	3～16
	0	1～3	すべて
	1	0～3	すべて
ディレクトリ	1	18	1～12
ID	1	18	13
FAT	1	18	14～16

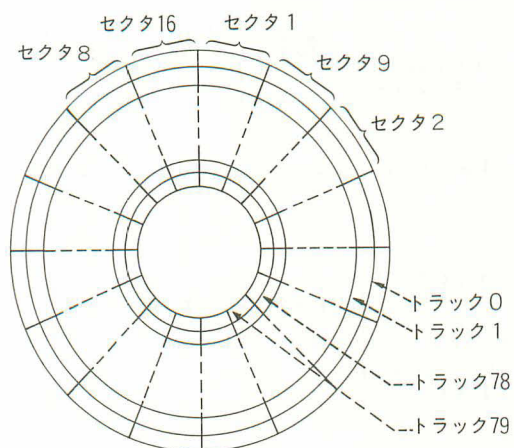
図2　5インチ両面倍密度フロッピィディスク

	サーフェス番号	トラック番号	セクタ番号
IPL	0	0	1～2
Disk コード	0	0	3～16
	0	1～8	すべて
	1	0～8	すべて
ディレクトリ	0	40	1～12
ID	0	40	13
FAT	0	40	14～16

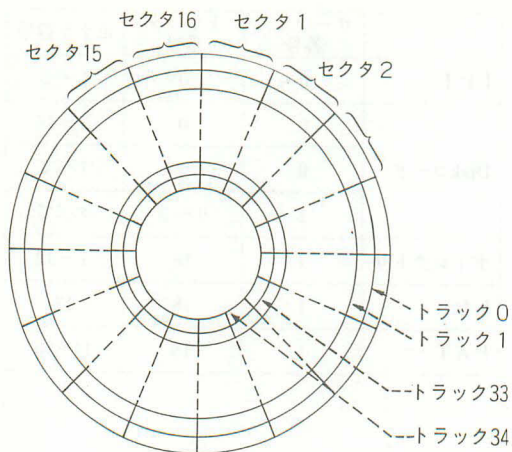
図3　5インチ両面倍密度倍トラックフロッピィディスク

	トラック番号	セクタ番号
IPL	0	1～2
Diskコード	0	3～16
	1～6	すべて
ディレクトリ	18	1～12
ID	18	13
FAT	18	14～16

図4　5インチ片面倍密度フロッピィディスク

	サーフェス番号	トラック番号	セクタ番号
IPLおよびシステム予約	0	0	すべて
Disk コード	0	1	すべて
	1～2	0～1	すべて
	3	0	すべて
ディレクトリ	0	75	すべて
	1～2	75	すべて
ID	3	75	1
FAT	3	75	2～33

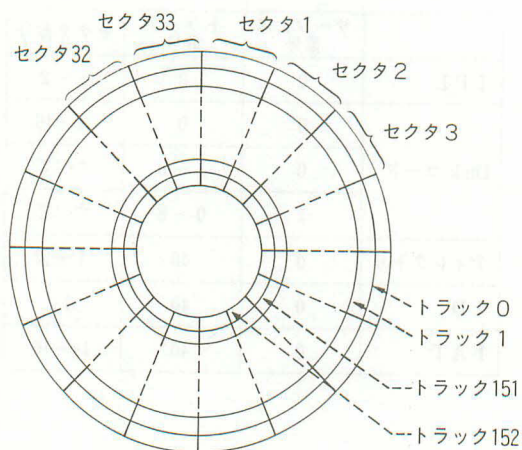



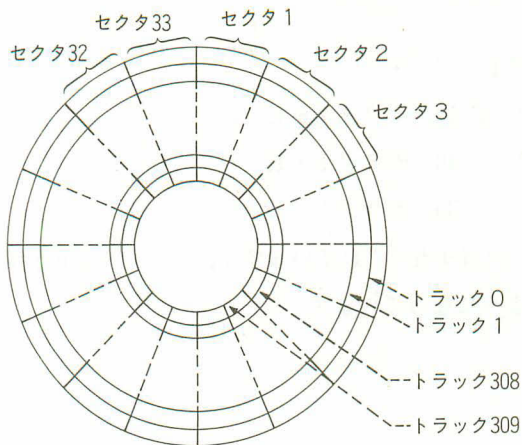
図5　5インチ固定ディスク（5MB）

	サーフェス番号	トラック番号	セクタ番号
IPLおよびシステム予約	0	0	すべて
Disk コード	0	1	すべて
	1～2	0～1	すべて
	3	0	すべて
ディレクトリ	0	150	すべて
	1～2	150	すべて
ID	3	150	1
FAT	3	150	2～33

図6　5インチ固定ディスク（10 MB）

7-1-2 ディスクアドレスとクラスタとの変換

1）5インチ片面のとき

〈クラスタ〉＝〈トラック〉×2＋〈セクタ〉¥9

〈トラック〉＝〈クラスタ〉¥2

〈セクタ〉＝（〈クラスタ〉MOD 2）×8＋1から8セクタ

2）5インチ両面のとき

〈クラスタ〉＝〈トラック〉×4＋〈サーフェス〉×2＋〈セクタ〉¥9

〈トラック〉＝〈クラスタ〉¥4

〈サーフェス〉＝（〈クラスタ MOD 4）¥2

〈セクタ〉＝（〈クラスタ〉MOD 2）×8＋1から8セクタ

3）5インチ倍トラック・8インチのとき

〈クラスタ〉＝〈トラック〉×2＋〈サーフェス〉
〈トラック〉＝〈クラスタ〉¥2
〈サーフェス〉＝〈クラスタ〉MOD 2
（以上3式 共通）

〈セクタ〉＝1セクタ～16セクタまで全部（5インチ倍トラック）

〈セクタ〉＝1セクタ～26セクタまで全部（8インチ）

7-1-3 ディレクトリ

ディレクトリには，ファイル名，ファイルの属性，ファイルが格納されている先頭クラスタ番号がしまわれています。これによりファイル名とファイルが格納されている場所との対応がつけられます。

ディレクトリの位置は，

5インチ片面………………トラック18，セクタ1～12

5インチ両面………………サーフェス1，トラック18，セクタ1～12

5インチ両面倍トラック…サーフェス0，トラック40，セクタ1～12

8インチ……………………サーフェス0，トラック35，セクタ1～22

となっています。1つのファイルに対して16バイトが割り当てられていますが，そのうち使用されているのは11バイトです。ディレクトリは下図のようになっています。

8インチ両面のとき

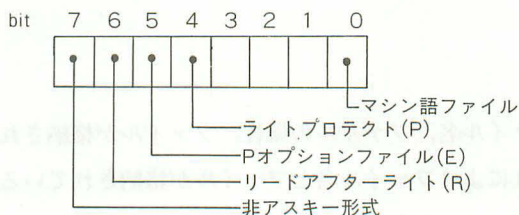
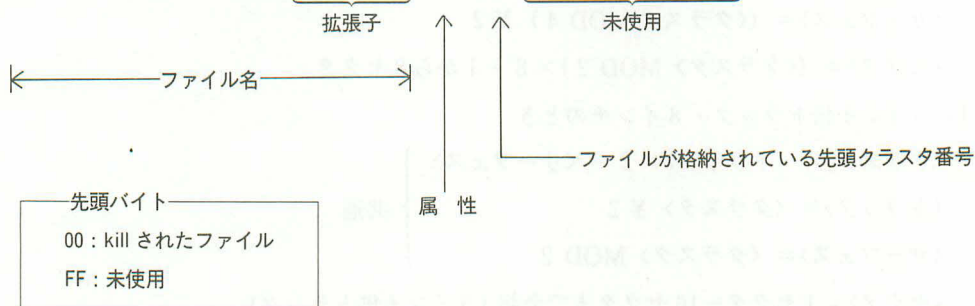
```
Dr 01,Sur 00,Tr 0023,Sec 01
0000 4B 41 4E 4A 49 20 44 49 43 00 47 FF FF FF FF FF      KANJI DIC G
0010 66 6F 72 6D 61 74 6E 69 70 80 45 FF FF FF FF FF      formatnip_E
0020 73 79 73 67 65 6E 6E 69 70 80 44 FF FF FF FF FF      sysgennip_D
0030 62 61 63 6B 75 70 6E 38 38 80 43 FF FF FF FF FF      backupn88_C
0040 78 66 69 6C 65 73 6E 38 38 80 41 FF FF FF FF FF      xfilesn88_A
0050 73 65 74 69 6E 66 6E 38 38 80 4C FF FF FF FF FF      setinfn88_L
0060 44 44 63 6F 6E 76 6E 38 38 80 40 FF FF FF FF FF      DDconvn88_@
0070 66 6F 72 6D 61 74 68 64 20 80 4D FF FF FF FF FF      formathd _M
0080 72 65 63 6F 76 20 68 64 20 80 3F FF FF FF FF FF      recov hd _?
0090 78 66 69 6C 65 73 68 64 20 80 4F FF FF FF FF FF      xfileshd _O
00A0 64 69 72 20 20 20 68 64 20 80 50 FF FF FF FF FF      dir   hd _P
00B0 00 65 6E 75 38 20 20 20 20 80 3C FF FF FF FF FF       enu8    _<
00C0 64 65 6D 6F 31 20 20 20 20 80 51 FF FF FF FF FF      demo1    _Q
00D0 64 65 6D 6F 32 20 20 20 20 80 3B FF FF FF FF FF      demo2    _;
00E0 64 65 6D 6F 33 20 20 20 20 80 53 FF FF FF FF FF      demo3    _S
00F0 64 65 6D 6F 35 20 20 20 20 80 54 FF FF FF FF FF      demo5    _T
```

拡張子　　未使用

|←―――ファイル名―――→|

属　性

ファイルが格納されている先頭クラスタ番号

先頭バイト
00：kill されたファイル
FF：未使用

7-1-4 IDセクタ

IDにはユーザーズマニュアルにある通り，ディスク全体の属性，一度にOPENできるファイルの数，電源ON（リセット）時に実行される文が格納されています。

IDは，

5インチ片面………………トラック18，セクタ13

5インチ両面………………サーフェス1，トラック18，セクタ13

5インチ両面倍トラック…サーフェス0，トラック40，セクタ13

8インチ………………………サーフェス0，トラック35，セクタ23

に割り当てられています。

8インチ両面の場合

```
Dr 01,Sur 00,Tr 0023,Sec 17
0000 00 03 43 4C 53 20 3A 20 46 49 4C 45 53 20 3A 20   CLS : FILES :
0010 50 52 49 4E 54 20 44 53 4B 46 28 31 29 3B 22 20  PRINT DSKF(1);"
0020 43 6C 75 73 74 65 72 73 20 66 72 65 65 22 00 00  Clusters free"
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

1度にOPENできるファイル数

属性（ディレクトリを参照）

なお，1度にOPENできるファイル数と電源ON時に実行される文は，システムディスクでないと意味を持ちません。

7-1-5 FAT（File Allocation Table）

FATはファイルの格納状態を示します。ファイルが1クラスタに納まらない場合，残りを別のクラスタに書き込まなければなりません。この「別のクラスタ」をどこにするかにはいろいろな方法がありますが，N_{88}-DISK BASIC (86) では，適当に空いているクラスタに書き込みます。このとき，どこのクラスタに書き込んだかを記録しておかないとあとで困ることになります。また，「別の

クラスタ」をさがす時に，どこが空きクラスタかが分からなければなりません。これらの情報を記録したものが，FATです。

ではこのFATはどのようになっているか見てみましょう。

FATの位置は，

5インチ片面………………トラック18，セクタ14～16

5インチ両面………………サーフェス1，トラック18，セクタ14～16

5インチ両面倍トラック…サーフェス0，トラック40，セクタ14～16

8インチ……………………サーフェス0，トラック35，セクタ24～26

となっていて，3つのセクタとも同じものが入っています。

8インチ両面の場合

```
Dr 01,Sur 00,Tr 0023,Sec 18          クラスタ4AH
0000 FE FE FE FE FE FE FF FF FF FF FF FF FF FF FF FF
0010 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF CB                  ヒ
0030 D6 C3 CB D4 C3 C3 C1 C1 37 C6 C8 CD CC C9 3D D3   ヨテヒヤテテチチ7ニネヘフノ=モ
0040 C2 40 D1 CF C6 44 FE C1 49 C1 4B 4C CC C7 4F CA   ツ@ムマニD チIチKLフヌOハ
0050 CB CB CB C9 CB D3 D0 C1 C3 CF C4 D3 D6 C1 C3 FF   ヒヒヒノヒモミチテマトモヨチテ
0060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0070 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0090 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

クラスタ4AHを見て下さい。これはCAT1というファイルの先頭クラスタです。ここには4BHという値が入っています。これはデータがクラスタ4AHに入り切れず，クラスタ4BHに続いていることを示します。クラスタ4BHを見ると4CHとなっています。クラスタ4CHを見ると値はCCHとなっています。クラスタCCHというものはありません。値がC1H～DAH(5インチの時はC1H～C8H，5インチ2DDの時はC1H～D0H）の時はここのクラスタでファイルが終わっていて，下位5ビット（8インチの時1～1AH，5インチの時1～8H，5インチ2DDの時は01H～10H）がそのクラスタで実際に使用しているセクタ数を表わします。ここではクラスタ4CHのうち12（CH）セクタを使用してファイルが終わっていることを示します。

これらをまとめると次のようになります。

バイトのデータ (16進)	クラスタの使用状態
8インチ両面の時　0 ～99 5インチ両面の時　0 ～9F 5インチ2DDの時　0 ～9F 5インチ片両の時　0 ～45	使用中。連続したクラスタの一部であり，後続するクラスタを持つ。値が，後続するクラスタの番号を示している。
8インチ両面の時　C1～DA 5インチ両・片面の時　C1～C8 5インチ2DDの時　C1～D0	使用中。連続したクラスタの最後のクラスタであり，下5 (5インチの時は下4) ビットの内容が，そのクラスタで実際に使われているセクタの数を表わす。
FE	予約ずみのクラスタで，ファイルとして使うことはできない。(DISKコード，IPL，ディレクトリ，FAT自身を含むクラスタがこれである。)
FF	未使用。

7-2 DSKF関数

DSKF関数は，ディスクファイルの構造に関する情報を返す関数で，機能を指定するによって，表7-2のような値が得られます。

機能番号	8インチ両面		5インチ両面		5インチ2DD		5インチ片面		機　　能
	16進	10進	16進	10進	16進	10進	16進	10進	
0	4C	76	27	39	4F	79	22	34	片面あたりの最大トラック番号
1	1A	26	10	16	10	16	10	16	1トラックあたりのセクタ数
2	01	1	01	1	01	01	00	0	片面……0，両面……1
3	01	1	02	2	01	01	02	2	1トラックあたりのクラスタ数
4	9A	154	A0	160	A0	160	46	70	クラスタの総数
5	23	35	12	18	28	40	12	18	ディレクトリが格納されているトラック番号
6	1A	26	08	8	10	16	08	8	1クラスタあたりのセクタ数
7	18	24	0E	14	0E	14	0E	14	FATの開始セクタ番号
8	1A	26	10	16	10	16	10	16	FATの終了セクタ数
9	03	3	03	3	03	3	03	3	FATの個数
10	17	23	0D	13	0D	13	0D	13	IDが格納されているセクタ番号

表7-2 DSKF機能一覧表

7-3 標準ディスク

7-3-1 フォーマッティング

フロッピィディスクを初期化することを「フォーマッティングする」と言います。フォーマッティングのしかたには2段階あり，ひとつは「物理フォーマット」で，もうひとつは「システムフォーマット」です。

① 物理フォーマット

フロッピィディスクにサーフェス，トラック，セクタなどのアドレスを割り当てることを物理フォーマットと言います。市販されている8インチ標準フロッピィディスクは通常IBMフォーマットと呼ばれる物理フォーマットが施されています。これは，PC-9881用にNECが供給しているPC-9884とは異なる物理フォーマットとなっています。

② インターリーブ13とは？

一般的なディスクは，トラック上にセクタが1から26まで順番に並んでいますが，インターリーブ13でフォーマットしたディスクはセクタが1つおきになっています。第1セクタの次が第14セクタ，その次が第2セクタ，その次が第15セクタ………と続いています。PC-9881を使用して連続したセクタを読み書きする際には，非常に高速になります。

あるトラックの第1セクタから第3セクタを続けて読む場合を例にとって考えます。ディスクのヘッドが第1セクタを読みとった後，第2セクタを読みに行くまでの間にはいろいろな処理が必要で，セクタが順に並んでいる場合には，第2セクタを読みとろうとした時にはすでに第2セクタの先頭がヘッドを行きすぎてしまい，第2セクタがぐるっとトラックを1周してくるまで待つ必要があります。

インターリーブ13のディスクでは，第1セクタの次は第14セクタですから，第14セクタが行きすぎるまで待てばその次の第2セクタが読めるという訳です。

従ってこの形式の方がずっと速いということになります。ただし，これはDisk BASICで読み書きする際にのみ当てはまります。

次に示すプログラムは，この「インターリーブ・フォーマット」を行うものです。ドライブ2にフォーマットするディスケットを入れて実行して下さい。この用途には次のようなことが考えられます。

- 市販の新しいディスケットを物理フォーマット。
- CP／M-86で使用していたディスケットをN_{88}-Disk BASIC(86)で使う場合に再フォーマット。
- その他のOSやディスクシステムで使用していたディスケットを再フォーマット。

インターリーブ13フォーマット

```
1 'save "IntLV"
100 ' ------------------------------
110 ' Interleave 13 Format Program
120 ' ------------------------------
130 CONSOLE 0,25,0,0:WIDTH 80,25
140 CLEAR ,&H1D00:DEF SEG=&H1D00:DEFINT A-Z:AD=0
150 PRINT "Interleave 13 Formatting :"
160 FOR I=0 TO &HCF:READ D$:POKE AD+I,VAL("&H"+D$):NEXT
170 FORM13=0:PRINT
180 PRINT "Place NEW diskette on DRIVE 2 and Press RETURN.";
190 LINE INPUT A$
200 INPUT "Sure (y/n) ";A$
210 IF A$="y" OR A$="Y" OR A$="ン" THEN 220 ELSE END
220 ' ------ Format Start ---------
230 PRINT "Now formatting ..."
240 CYL=0:SF=0:DW=0:GOSUB *FORM.TR:IF STS<>0 THEN *FORM.ERR
250 FOR I=1 TO 76
260   CYL=I:SF=0:DW=1:GOSUB *FORM.TR:IF STS<>0 THEN *FORM.ERR
270 NEXT
280 FOR I=0 TO 76
290   CYL=I:SF=1:DW=1:GOSUB *FORM.TR:IF STS<>0 THEN *FORM.ERR
300 NEXT
310 PRINT "Interleave 13 Format Complete.":END
320 ' ------- Call Sub --------
330 *FORM.TR
340   CALL FORM13(STS,CYL,SF,DW)
350 RETURN
360 *FORM.ERR
370   PRINT "Format Error : Track=";CYL;" Surface=";SF:END
380 ' ------ Machine Code -------
390 DATA 8B,4F,0A,8E,C1,8B,77,08,26,8A,24,8B,4F,06,8E,C1
400 DATA 8B,77,04,26,8A,34,8B,4F,02,8E,C1,8B,37,26,8A,14
410 DATA 53,8A,DC,0E,07,BD,68,00,B9,1A,00,8B,F5,26,88,1C
420 DATA 26,88,74,01,26,88,54,03,83,C6,04,E0,F0,8A,C2,0A
430 DATA C0,75,05,B8,91,1D,EB,03,B8,91,5D,8A,CB,8A,EA,BB
440 DATA 68,00,B2,40,CD,1B,5B,72,03,B8,00,00,8B,4F,0E,8E
450 DATA C1,8B,77,0C,26,89,04,CF,00,00,01,00,00,00,0E,00
460 DATA 00,00,02,00,00,00,0F,00,00,00,03,00,00,00,10,00
470 DATA 00,00,04,00,00,00,11,00,00,00,05,00,00,00,12,00
480 DATA 00,00,06,00,00,00,13,00,00,00,07,00,00,00,14,00
490 DATA 00,00,08,00,00,00,15,00,00,00,09,00,00,00,16,00
500 DATA 00,00,0A,00,00,00,17,00,00,00,0B,00,00,00,18,00
510 DATA 00,00,0C,00,00,00,19,00,00,00,0D,00,00,00,1A,00
```

③ システムフォーマット

物理フォーマットを行うと，サーフェス，トラック，セクタが割り当てられるため，N_{88}-Disk BASIC(86)のDSKI$，DSKO$で読み書きができるようになります。例えば，ドライブ1のサーフェス1，トラック2，セクタ3を読むには，次のようにします。

```
10 FIELD#0,128 AS A$,128 AS B$
20 DM$=DSKI$(1,1,2,3)
30 PRINT A$;B$
40 END
```

しかし，物理フォーマットだけでは，N_{88}-Disk BASIC(86)のSAVE，LOAD，ファイル処理(OPEN，PRINT # 1，GET # 1，PUT # 1，CLOSE)などができません。これは，N_{88}-Disk BASIC(86) がそれらの処理を行ううえで必要な管理情報がないためです。この管理情報を記録するために，FAT，ディレクトリ，IDの初期化が必要で，これをシステムフォーマットと呼びます。これは物理フォーマットに対して，ソフト的なフォーマットです。

PC-9884に入っているformat.n 88のプログラムは，このソフト的なフォーマットを行うものです。

④ サンプル・プログラム

N_{88}-Disk BASIC(86)でインターリーブ・フォーマットしたものと通常のものを比較してみると，読み書きにかなりの時間の差がみられます。次にデータをシーケンシャルファイルとランダムファイルに読み書きするプログラム例をあげます。

ランダムファイル

"A"の文字列128文字と"B"の文字列128文字を1つのデータとして，ランダムファイルに書き出し，読み込みます。データの数は，RUNしたときに入力してください。

```
1 'save "INT13.RND"
100 '----- Prepare Data --------
110 A1$=STRING$(128,"A")
120 B1$=STRING$(128,"B")
130 INPUT "No of data  ";MX
140 '----- Write Data ---------
150 TIME$="00:00:00"
160 OPEN "2:demo2" AS #1
170 FIELD #1,128 AS A$,128 AS B$
180 LSET A$=A1$
190 LSET B$=B1$
200 FOR I=1 TO MX
210   PUT #1,I
220 NEXT I
230 CLOSE #1
240 PRINT TIME$
250 '----- Read Data ----------
260 TIME$="00:00:00"
270 OPEN "2:demo2" AS #1
280 FIELD #1,128 AS A$,128 AS B$
290 FOR I=1 TO MX
300   GET #1,I
310 NEXT I
320 CLOSE #1
330 PRINT TIME$
340 END
```

シーケンシャルファイル

"A"の文字列255文字を1つのデータとして，シーケンシャルファイルに書き出し，読み込みます。データの数は，RUNしたときに入力してください。

```
1 'save "INT13.SEQ"
100 '----- Prepare Data --------
110 A$=STRING$(255,"A")
115 INPUT "No. of data ";MX
120 '----- Write Data ---------
130 TIME$="00:00:00"
140 OPEN "2:demo1" FOR OUTPUT AS #1
150 FOR I=1 TO MX
160 PRINT #1,A$
170 NEXT I
180 CLOSE #1
190 PRINT TIME$
200 '----- Read Data ----------
210 TIME$="00:00:00"
220 OPEN "2:demo1" FOR INPUT AS #1
230 FOR I=1 TO MX
240 INPUT #1,A$
250 NEXT I
260 CLOSE #1
270 PRINT TIME$
280 END
```

⑤ 読み書きテスト結果

ランダム

通　　常		インターリーブ13	倍　率
データ数　100			
WRITE	17秒	16秒	1.06
READ	17秒	1秒	17.00
データ数　200			
WRITE	34秒	19秒	1.79
READ	34秒	3秒	11.33
データ数　300			
WRITE	51秒	22秒	2.32
READ	51秒	5秒	10.20

シーケンシャル

通　　常		インターリーブ13	倍　率
データ数　100			
WRITE	43秒	26秒	1.65
READ	17秒	18秒	
データ数　200			
WRITE	1分27秒	55秒	1.58
READ	34秒	36秒	
データ数　300			
WRITE	2分13秒	1分24秒	1.58
READ	52秒	54秒	

プログラムのSAVE・LOAD

通　　常		インターリーブ13	倍　率
4クラスタのプログラム（約20K）			
SAVE	34秒	14秒	2.42
LOAD	14秒	1秒	14.00

7-3-2 ディスクBIOSコマンド

8インチフロッピィディスクは，μPD 765 Aフロッピィディスクコントローラ（FDC）により制御されています。データ転送は，DMA（Direct Memory Access）によって行われ，フロッピィディスクに対するデータの読み書きはDMA制御部を用いて行われます。

この節では，8インチ・ディスクのBIOS（基本入出力）コマンドとその使い方を説明します。これらのコマンドはμPD 765 Aを直接コントロールしていますので，N_{88}-Disk BASIC（86）では困難な，あるいは不可能な処理ができるようになります。

① 概　　要

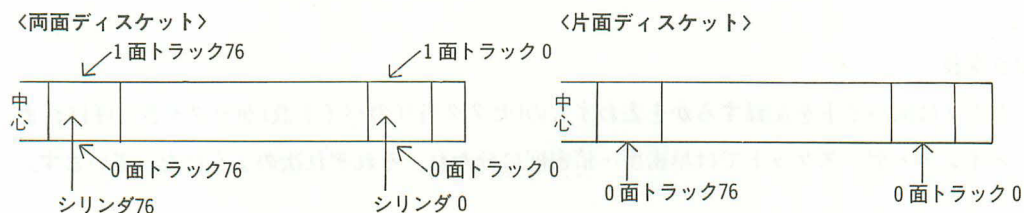
まず，通常ユーザーが使うBIOSコマンドの一覧を示します。

8インチ・ディスクBIOSコマンド一覧表

No.	コマンド名	機　　能
1	リードＩＤ	トラック上のエラーのないIDを読み取る
2	リードデータ	ディスク上のデータを読み取る
3	ライトデータ	ディスク上にデータを書き込む
4	シーク	シリンダにアームを移動しヘッドを選択する
5	フォーマット	1トラック分のセクターフォーマッティングを行う

次に，ディスクBIOSコマンドを使用するうえでの用語とコマンドの一般形式について説明します。

● シリンダとヘッド

シリンダとは，両面ディスケット上の0～76までのトラックのことをいいます。
両面ディスクのラベルを貼る面は1面であり，反対側は0面となっています。なお片面ディスケットの場合は0面しか使いません。このことを図で表わすと次のようになります。

ヘッドは，データを読み書きするもので，ヘッド番号は両面ディスケットでは0か1（0面か1面）であり，片面ディスケットでは常に0です。

このシリンダとヘッドにより，どの物理トラックをアクセスするかが決定されます。これは，N_{88}-Disk BASIC(86)のサーフェス(面)とトラックによる指定と同じです（次表参照)。

サーフェス・トラックとヘッド・シリンダの対応表

N_{88}-Disk BASIC (86)			ディスク BIOS	
クラスタ番号	サーフェス	トラック	ヘッド	シリンダ
0	0	0	0	0
1	1	0	1	0
2	0	1	0	1
3	1	1	1	1
4	0	2	0	2
5	1	2	1	2
・ ・	・ ・	・ ・	・ ・	・ ・
150	0	75	0	75
151	1	75	1	75
152	0	76	0	76
153	1	76	1	76

● セクタ長

1セクタに何バイトを記録するかを表わすもの(セクタ当りのバイト数)がセクタ長と呼ばれます。8インチ・ディスケットでは単密度・倍密度に分かれ，それぞれ次のようになっています。

	セクタ長指定	セクタ当りのバイト数	最終セクタ	備　　考
単密度	00	128バイト	26	IBM1型
	01	256バイト	15	N_{88}-Disk BASIC(86)(0ヘッド，0シリンダのみ)
	02	512バイト	8	
倍密度	01	256バイト	26	IBM2D型, CP／M-86, N_{88}-Disk BASIC(86)
	02	512バイト	15	IBM PC準拠
	03	1024バイト	8	MS-DOS

これを簡単にまとめると下図のようになります。

セクタ長指定	00	01	02	03
セクタ長(バイト／セクタ)	128	256	512	1024

● ID 情報

ディスク媒体には物理フォーマット時に各セクターを物理的に識別するための領域が書き込まれており，この領域を ID と呼びます。ID には，シリンダ番号，ヘッド番号，セクタ番号，セクタ長が記録されています。これらをあわせて ID 情報といいます。

● デバイス種別・ユニット番号

フロッピィディスクの種別（タイプ）とユニット番号（ドライブ番号）は次のようになっています。

デバイスタイプ	ドライブ番号			
	1	2	3	4
8インチ	90H	91H	92H	93H
5インチ	50H	51H	52H	53H
5インチ2DD	70H	71H	72H	73H

● データ長

実際に読み書きするデータの長さ（バイト単位）をデータ長といいます。

長い前置きでしたが，以上のことを理解しているとディスク BIOS コマンドが楽々と使えます。さて，いよいよ各コマンドの使い方に入ります。

② リード ID

ID 情報の読み出しを行います。これは指定されたトラック上で最初に正常に読み取れた ID を ID 情報として対応レジスタに格納します。これにより，各トラックがどういうフォーマットになっているかが分かります。各レジスタには次のようにセットし，インタラプトコール（INT 1BH）を行います。

```
AH ←── 5AH（倍密度）
        1AH（単密度）
AL ←── 90H～93H（ドライブ1～4）
CL ←── 0～76 （シリンダ番号）
DH ←── 0～1   （ヘッド番号）
```

正常終了すれば，CF が 0 となり次の各レジスタに ID 情報が格納されます。

ＣＬ ←―― シリンダ番号

ＣＨ ←―― セクタ長

ＤＬ ←―― セクタ番号

ＤＨ ←―― ヘッド番号

異常終了すれば，CFが１となりAHに次のようなエラーステータスが入ります。

ＡＨの内容	エ ラ ー の 内 容
４０Ｈ	デバイスが異常
６０Ｈ	ユニットがレディの状態でない
Ｃ０Ｈ	正しいＩＤが見つからない
Ｅ０Ｈ	正しいＩＤが見つからず，しかもＩＤアドレスマークが１回も検出されなかった

それでは，N_{88}-Disk BASIC (86) のディスケットを対象にして具体例をあげましょう。

● ０ヘッド，０シリンダのIDを読む

このトラックは単密度で，セクタ長128バイトになっています。ドライブ１にディスケットがあるものとします。

```
0000 B41A        MOV    AH,1A:  単密
0002 B090        MOV    AL,90:  ドライブ1
0004 B100        MOV    CL,00:  シリンダ0
0006 B600        MOV    DH,00:  ヘッド0
0008 CD1B        INT    1B   :  コール
000A CF          IRET
```

これはモニタモードで入力して，実行してみて下さい。Xコマンドで各レジスタの値とフラグの内容が一目で分かります。

結果は次のようになります。

ＣＬ＝0　　；　シリンダ0

ＣＨ＝0　　；　セクタ長（128バイト）

ＤＬ＝1　　；　1セクタ

ＤＨ＝0　　；　ヘッド0

CF，AH も 0 で正常終了しています。

こんどは単密モードでヘッド 1，シリンダ 0 を読んでみましょう。

MOV　DH，01 H とするだけで良いですね。

すると，CF が 1 となり，AH に E 0 H が入ります。正常に読めていないことが分かります。ここは倍密になっているためです。そこで，MOV　AH，5 AH としてみましょう。これなら正常終了します。そして，CH の値が 1 となり，セクタ長が 256 バイトであることを表わしています。

このリード ID で，IBM フォーマットや CP／M-86，MS-DOS などのディスケットの ID 情報を読んでみるとそれぞれのフォーマットが分かります。

③　リードデータ

指定ドライブの任意の開始セクタからメモリ領域に指定された長さ（データ長，バイト単位）だけデータを読み出します。

各レジスタを次のようにセットし，インタラプトコール（INT　1 BH）を行います。

AH ←―― 5 6 H（倍密度）
　　　　　1 6 H（単密度）
AL ←―― 9 0 ～ 9 3 H（ドライブ 1 ～ 4）
BX ←―― リードするデータ長（バイト数）
CL ←―― 0 ～ 7 6（シリンダ番号）
DH ←―― 0 ～ 1　（ヘッド番号）
DL ←―― 1 ～ 2 6（開始セクタ）
CH ←―― 0 ～ 3　（セクタ長，128～1024）
ES：BP ←―― データ格納の先頭アドレス（偶数番地にすること）

正常終了すれば CF が 0 となり，ES（セグメントアドレス），BP（オフセットアドレス）が示すデータバッファ領域先頭アドレスからデータ長分だけデータが格納されます。

異常終了すれば，CF が 1 となり AH に次ページのようにエラーステータスが入ります。

AHの内容	エラーの内容
20H	メモリアドレスがバンクにまたがっているか，奇数番地からはじまるように指定している
30H	1回の転送容量を越えてデータ長を指定している
40H	デバイス異常
50H	一定時間内にデータ転送を終了できない
60H	ユニットがレディ状態ではない
A0H	ID読み出し時にCRCエラーが発生した
C0H	トラック内に指定されたセクタが見つからない
E0H	上記に加え，IDが1個も見つからない

N_{88}-Disk BASIC(86)の8インチのIPLを読んでみることにして具体例を示しましょう。IPLは，ヘッド0，シリンダ0，セクタ1～4に入ってることになっています（実際はセクタ1だけにしか入っていません)。

```
0000 B80018        MOV    AX,1800
0003 8EC0          MOV    ES,AX   ; セグメント1800H
0005 BD0000        MOV    BP,0000 ; オフセット0
0008 B416          MOV    AH,16   ; 単密
000A B090          MOV    AL,90   ; ドライブ1
000C BB0002        MOV    BX,0200 ; 128バイト×4セクタ
000F B100          MOV    CL,00   ; シリンダ0
0011 B600          MOV    DH,00   ; ヘッド0
0013 B201          MOV    DL,01   ; 開始セクタ1
0015 B500          MOV    CH,00   ; セクタ長128バイト
0017 CD1B          INT    1B
```

上記を実行後，セグメント1800H，オフセット0Hから逆アセンブルしてみるとIPLがどういうことをしているかが分かると思います。

次に本当にリードしたかどうかが一目で分かるところを読んでみましょう。それはディレクトリです。これは，ヘッド0，シリンダ35（23H)，セクタ1～22にあります。

変更する点は次のとおりです。

```
MOV    AH,56   ; 倍密
MOV    BX,1600 ; 256バイト×22セクタ
MOV    CL,23   ; シリンダ35（10進数)
MOV    CH,01   ; セクタ長256バイト
```

モニターモードでC 1800〔RET〕，E 0〔RET〕とすると，ディレクトリがちゃんと転送されていることが分かります。

④　ライトデータ

ドライブ，開始セクタを指定し，データバッファ領域の先頭アドレスとデータ長を指定すれば，その内容を指定のディスクアドレスに書き込みます。

各レジスタを次のようにセットし，インタラプトコール（INT 1BH）を行います。

AH ←―― 5 5 H　（倍密度）
　　　　　1 5 H　（単密度）
AL ←―― 9 0 H～9 3 H（ドライブ1～4）
BX ←―― ライトするデータ長（バイト数）
CL ←―― 0～7 6（シリンダ番号）
DH ←―― 0～1　（ヘッド番号）
DL ←―― 1～2 6（開始セクタ）
CH ←―― 0～3　（セクタ長，128～1024）
ES：BP ←―― データバッファ領域の先頭アドレス（偶数アドレスにすること）

CFが0で正常終了，1で異常終了となります。異常の場合，AHにはリードデータのときと同じエラーステータスが入りますが，1つ追加されています。

AHの内容	エ ラ ー の 内 容
7 0 H	ライトプロテクトがかかっていた

それではサンプルとして，アスキーコード0～255の256バイトをヘッド1，シリンダ0の1セクタ目に書き込んでみましょう。ここは未使用になっている部分です。このテストには壊れても良いようなディスケットを用いて下さい。万一，大切なデータが消えることにもなりかねませんので。

```
0000 B80018          MOV    AX,1800
0003 8EC0            MOV    ES,AX   ; セグメント1800H
0005 BD0000          MOV    BP,0000; オフセット0
0008 30C0            XOR    AL,AL   ; AL=0
000A B90001          MOV    CX,0100; 256バイトのデータ
000D 31DB            XOR    BX,BX   ; BX=0
000F 26              ES:
0010 8807            MOV    [BX],AL; AL=0~255
0012 FEC0            INC    AL      ; アスキーコードを格納
0014 43              INC    BX
0015 E2F8            LOOP   000F
0017 B455            MOV    AH,55   ; 倍密
0019 B090            MOV    AL,90   ; ドライブ1
001B BB0001          MOV    BX,0100; 256バイト書き込み
001E B100            MOV    CL,00   ; シリンダ0
0020 B601            MOV    DH,01   ; ヘッド1
0022 B201            MOV    DL,01   ; 開始セクタ
0024 B501            MOV    CH,01   ; セクタ長256バイト
0026 CD1B            INT    1B      ; コール
```

⑤ シーク

ドライブとシリンダ物理番号を指定するとそこまで，読み書きヘッドをシーク（移動）させます。前出のリード・ライト用のBIOSコマンドはシーク動作を同時に行うことができるようになっており，使用例はすべてシーク動作も行うように指定しています。BIOSコマンド識別コードの5ビット目（10 H）を1にするとシーク動作を行い，0にすると現在のシリンダを対象とした動作を行います。

各レジスタを次のようにセットし，インタラプトコール（INT 1BH）します。

AH ←── 10H	;	5ビット目をオン
AL ←── 90H~93H	;	ドライブ1~4
CL ←── 0~76	;	シリンダ番号

終了条件は前出のもとのと同じです。

次にディスクのFATのところにシークする例をあげます。FATはヘッド0，シリンダ35(23H）にあります。

```
0000 B410            MOV    AH,10  ; シーク動作
0002 B090            MOV    AL,90  ; ドライブ1
0004 B123            MOV    CL,23  ; シリンダ
0006 CD1B            INT    1B     ; コール
```

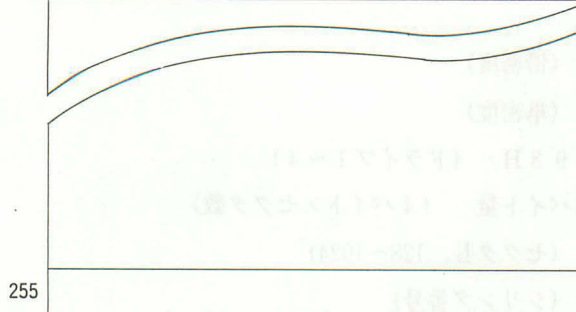
なお，シリンダ0へシークするコマンドも用意されています。

```
0000 B407        MOV    AH,07 ; シリンダ0へシーク
0002 B090        MOV    AL,90 ; ドライブ1
0004 CD1B        INT    1B    ; コール
```

このシーク動作は，シリンダ物理番号0の方向へ，1シリンダずつシークし，トラック0信号を検出するまでくり返します。

⑥ フォーマット

1トラック分のセクタ長，トラック当たりのセクタ数，データ部に書き込むデータパターンなどに従って物理フォーマッティングを行います。

各レジスタを次のようにセットし，インタラプトコール（INT 1BH）します。

AH ←── 5DH （倍密度）
　　　　 1DH （単密度）
AL ←── 90H～93H （ドライブ1～4）
BX ←── データのバイト量 （4バイト×セクタ数）
CH ←── 0～3 （セクタ長，128～1024）
CL ←── 0～76 （シリンダ番号）
DH ←── 0～1 （ヘッド番号）
DL ←── データパターン （通常40H〝@″）
ES：BP ←── データバッファ領域先頭アドレス

セクタのID部にデータを書き込むときには次のようにシリンダ，ヘッド，セクタ，セクタ長の4バイトの情報をトラックのセクタ数分だけ用意しておく必要があります。

セクタシーケンスに従った論理セクタ番号

(i) 26セクタ／トラックの場合

物理セクタ番号(16) ＼ セクタシーケンス(16)	01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A
00，01 CP／M-86	物理セクタ番号と同じ
0DH BASIC	インターリーブ13 01 0E 02 0F 03 10 04 11 05 12 06 13 07 14 08 15 09 16 0A 17 0B 18 0C 19 0D 1A

(ii) 8セクタ／トラックの場合

物理セクタ番号(16) ＼ セクタシーケンス(16)	01 02 03 04 05 06 07 08
00，01 MS-DOS	物理セクタ番号と同じ

それでは，ドライブ2にあるディスケットのシリンダ0，ヘッド1の部分（未使用）をフォーマットする例を示します。フォーマットは，単密度で1セクタ512バイト，1トラック8セクタにするとします。

```
0000 B41D          MOV    AH,1D   ; 単密
0002 B091          MOV    AL,91   ; ドライブ2
0004 BB2000        MOV    BX,0020; 4バイト×8セクタ
0007 B502          MOV    CH,02   ; 512バイト／セクタ
0009 B100          MOV    CL,00   ; シリンダ0
000B B601          MOV    DH,01   ; ヘッド1
000D B240          MOV    DL,40   ; データパターン"@"
000F 0E            PUSH   CS
0010 07            POP    ES      ; セグメント
0011 BD2000        MOV    BP,0020; オフセット
0014 CD1B          INT    1B
```

```
シリンダ┐ ヘッド  セクタ番号┌セクタ長
0020 00 01 [01] 02|00 01 02 02|00 01 03 02|00 01 [04] 02
0030 00 01 [05] 02|00 01 06 02|00 01 07 02|00 01 [08] 02
                                                  └8セクタ
```

なお，インターリーブ・フォーマットの節で紹介したプログラムのソースリスト(付録)を参照されるとより具体的な使用例が分かると思います。

7-4　5インチ・ディスク

7-4-1 概　　要

PC-9801の5インチフロッピィディスク・インターフェイスは，PC-8001，PC-8801と同一インターフェイスで，デバイスもPC-8031-1 W，PC-8031-1 V，PC-8031-2 W,PC-80 S 31を接続することができます。

これらのデバイスはμPD 780およびROM (2 K)，RAM (16 K) を内蔵したインテリジェント型であり，PC-9801とは互いにμPD 8255を介してデータのやりとりを行います。

以下，5インチ・ディスクのBIOSコマンドとその使い方について説明します。

7-4-2 ディスクBIOSコマンド

① センス

デバイスの指定ドライブの状態をステータスとして通知します。AHにコマンド識別コード04Hを入れてドライブ番号を指定し，INT　1BHを行います。

AH ←—— 04H　(センス)
AL ←—— 50H～53H (ドライブ1～4)

正常終了すればCFが0となり，AHにステータス情報が次のように格納されます。

AHの内容	ステータスの意味
X0	片面装置
X2	両面装置・片面アクセスモード
X3	両面装置・両面アクセスモード
1X	ライトプロテクト（シールでの）

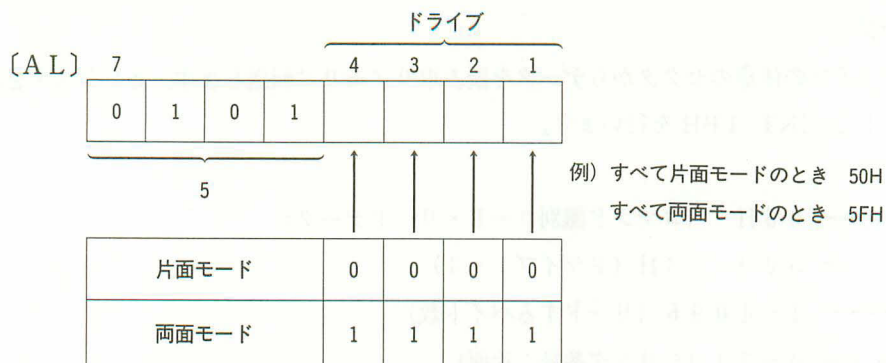
注）Xは下位・上位の4ビットのことで無視する。

異常終了すればCFが1となり，AHが4Xでデバイス異常を示します。
次に具体例としてあるディスクのドライブ1のステータスを調べてみましょう。

```
MOV   AH,  04H  ;  センス
MOV   AL,  50H  ;  ドライブ1
INT   1BH       ;  コール
```

実行後，CF＝0でAH＝03Hとなっていました。このディスクは両面装置で両面アクセスモードになっていることが分かります。ちなみに，プロテクトシールを貼って実行すると，AH＝13Hとなり，ライトプロテクトを検知しています。また，ディスクの電源をOFFにして実行すると，CF＝1でAH＝40Hとなり異常であることになります。

② イニシャライズ

デバイスのコントローラの初期設定を行います。このイニシャライズを行わないと他のコマンド（センスを除く）は起動できません。AHにコマンド識別コード03Hを入れ，ALには次のように設定してINT 1BHとします。

ドライブ1～4をすべて片面モードにするにはALに50Hを入れ，すべて両面モードにするには5FHを入れます。またドライブ1・2を両面に3・4を片面にするには，ALに53Hを入れます。なお，このイニシャライズコマンドにはステータス表示はありません。

では両面装置1台を使ってドライブ1を両面，ドライブ2を片面モードにイニシャライズしてみましょう。

```
MOV   AH,  03H  ;  イニシャライズ
MOV   AL,  51H  ;  ドライブ1→1，ドライブ2→0
INT   1BH       ;  コール
```

では，ドライブ1をセンスしてみます。

```
MOV   AH,  04H  ;  センス
MOV   AL,  50H  ;  ドライブ1
INT   1BH       ;  コール
```

結果はAHが03Hで両面装置の両面アクセスモードとなっています。こんどはドライブ2をセンスしてみます。

```
MOV   AL,  51H  ;  ドライブ2
```

結果はAHが02Hで両面装置の片面アクセスモードとなっています。無事イニシャライズされていたことになります。

③ リードデータ

指定したドライブの任意のセクタからデータを読み取りメモリに転送します。各レジスタを次のようにセットし，INT 1BHを行います。

AH ←── 06H (コマンド識別コード・リードデータ)
AL ←── 50H～53H (ドライブ1～4)
BX ←── 1～4096 (リードするバイト数)
CL ←── 0～34 (シリンダ番号：片面)
　　　　　0～39 (シリンダ番号：両面)
DH ←── 0～1 (ヘッド番号)
DL ←── 1～16 (セクタ番号)
ES：BP ←── データバッファ先頭アドレス

正常終了すればCFが0となりES：BPが示すアドレスからデータが格納されます。
異常終了すればCFが1となり，AHにステータス情報が次のように入ります。

AHの内容	ステータスの意味
4X	デバイス異常
2X	データバッファアドレスがバンクにまたがっているか奇数番地から始まるように指定している
9X	一定時間内にデータを転送できない
8X	リードエラー

注）Xは下位4ビットのことで無視する。

それではN88-Disk BASIC (86) の5インチ両面のIPLを読んでみましょう。IPLは，ヘッド0，シリンダ0，セクタ1～2に入っています。

```
0000 B406        MOV    AH,06   ;  リードデータ
0002 B050        MOV    AL,50   ;  ドライブ1
0004 BB0002      MOV    BX,0200 ;  2セクタ分リード
0007 B100        MOV    CL,00   ;  シリンダ0
0009 B600        MOV    DH,00   ;  ヘッド0
000B B201        MOV    DL,01   ;  セクタ1
000D 0E          PUSH   CS
000E 07          POP    ES      ;  ES=CS
000F BD2000      MOV    BP,0020 ;  オフセット20H
0012 CD1B        INT    1B      ;  コール
```

今度はディレクトリを読んでみましょう。ディレクトリは，ヘッド1，トラック18，セクタ1～12にあります。変更箇所は次のとおりです。

```
MOV     BX,0C00;  256×12バイトリード
MOV     CL,12  ;  シリンダ18
MOV     DH,01  ;  ヘッド1
```

④　ライトデータ

指定したドライブの任意のセクタにメモリ上のデータを書き込みます。各レジスタを次のようにセットし，INT　1BHを行います。

```
AH ←── 05H          (ライトデータ)
AL ←── 50H～53H     (ドライブ1～4)
BX ←── 1～4096      (ライトするバイト数)
CL ←── 0～34        (シリンダ番号：片面)
       0～39        (シリンダ番号：両面)
DH ←── 0～1         (ヘッド番号)
DL ←── 1～16        (セクタ番号)
ES：BP ←── データバッファ先頭アドレス
```

正常終了すればCFが0となりES：BPが示すアドレスからのデータが書き込まれます。異常終了すればCFが1となり，AHにそのステータスが格納されます。それは③リードデータと同じです。

PC-9801のテキストVRAMをディスクにライトしてみましょう。まずはその表示エリア（セグメントA000H，オフセット0～FFEH)です。書き込むところは，ヘッド0，シリンダ4からです。

```
0000 B405         MOV    AH,05   ;  ライトデータ
0002 B050         MOV    AL,50   ;  ドライブ1
0004 BBFF0F       MOV    BX,0FFF;   4095バイト
0007 B104         MOV    CL,04   ;  シリンダ4
0009 B600         MOV    DH,00   ;  ヘッド0
000B B201         MOV    DL,01   ;  セクタ1
000D 50           PUSH   AX      ;  AXを保存
000E B800A0       MOV    AX,A000
0011 8EC0         MOV    ES,AX   ;  ES＝A000H
0013 58           POP    AX
0014 BD0000       MOV    BP,0000;   オフセット0
0017 CD1B         INT    1B      ;  コール
```

次にアトリビュートエリア（セグメントA000H，オフセット2000H～2FFEH）です。

変更する部分は次のとおりです。

```
MOV      CL,15   ;   シリンダ21（10進数）
MOV      BP,2000;    オフセット2000H
```

このルーチンは，テキストVRAMの内容をディスクにセーブしておくときに便利です。なお，AHの05Hを06Hに変えてリードして実際に書き込まれているか確認してみて下さい。

⑤　フォーマット

指定ドライブに入っているフロッピィディスケットをフォーマッティングします。これにより，IDのセクタシーケンスは01Hデータ部はすべてFFHが書き込まれます。このコマンドの実行方法は次のとおりです。

```
AH ←── 0DH        ;  フォーマット
AL ←── 50～53H    ;  ドライブ1～4
INT   1BH         ;  コール
```

正常・終了条件は③リードデータと同じですが，エラーステータスの2X，9Xはありません。ドライブ2のディスケットをフォーマットしてみます。

```
MOV      AH,0D;   フォーマット
MOV      AL,51;   ドライブ2
INT      1B   ;   コール
```

⑥　片面・両面アクセス

両面装置のディスクに対して片面アクセスか両面アクセス（オペレーションモードという）の指示を行います。

AHに0EHを入れ，ALは②イニシャライズと同じ設定をします。ドライブ1を両面アクセスモード，ドライブ2を片面アクセスモードにするには次のようにします。

```
MOV      AH,0E;   セットオペレーションモード
MOV      AL,51;   ドライブ1－1，ドライブ2－0
INT      1B   ;   コール
```

なお，PC-9801Fの5インチ2DDについては，8インチのBIOSコールと同じで，ドライブ指定 1～4が70H～73Hとなるだけです。

7-5 ディスク・ユーティリティ・プログラム

ディスクの章のまとめとして,いままでの情報をプログラムに反映したディスク・ユーティリティをいくつか紹介いたします。

7-5-1 ファイルインフォメーション

ディスケットにセーブされているすべてのファイルの情報を出力します。これにより，ファイル名，属性，マシン語ファイルのアドレスおよびクラスタロケーションとリンクの各情報が得られます。なお，属性は次のような略語を用いています。

BAS………BASIC ファイル
ASC………アスキーファイル
MAC………マシン語ファイル
WTP………ライトプロテクト（Write Protect）
RAW………リードアフターライト（Read After Write）
PRT………プロテクト（Protect＝P オプション）

ファイルインフォメーション

```
1 'save "Finfo.n88
100 'File Information
110 DEFINT A-Z:CONSOLE 0,25,0,0
120 WIDTH 80,25:CLS:PRINT "■■■  File Information   ■■■"
130 LOCATE 0,1:INPUT "CRT (c) or Printer(p) ";CP$
140 IF CP$="c" OR CP$="C" THEN F$="SCRN:" :GOTO *INPT
150 IF CP$="p" OR CP$="P" THEN F$="LPT1:" :GOTO *INPT
160 LOCATE 0,1:PRINT SPACE$(35):BEEP:GOTO 130
170 '
180 *INPT
190 INPUT "Drive No. ";DN$:DN=VAL(DN$):D=15:MCL=DSKF(DN,4)-1
200 DIM FI$(D),EX$(D),AT$(D),CL$(D),FA(MCL)
210 OPEN F$ FOR OUTPUT AS #1
220 FOR I=0 TO 15
230 FIELD #0,I*16 AS DM$,6 AS FI$(I),3 AS EX$(I),
                            1 AS AT$(I),1 AS CL$(I)
240 NEXT I
250 '
260 MAX=DSKF(DN,0)
270 IF MAX=76 THEN DM$=DSKI$(DN,0,35,24) '-- 8 inch
280 IF MAX=39 THEN DM$=DSKI$(DN,1,18,14) '-- 5 inch DS
290 IF MAX=79 THEN DM$=DSKI$(DN,0,40,14) '-- 5 inch 2DD
300 IF MAX=34 THEN DM$=DSKI$(DN,18,14)   '-- 5 inch SS
310 '
320 FOR I=0 TO MCL
330    FA(I)=ASC(MID$(DM$,I+1,1))
340    IF FA(I)=255 THEN FR=FR+1
350 NEXT I
360 '
```

```
370 PRINT : NN=0
380 PRINT #1,"====== DRIVE ";DN; " =======";
390 PRINT #1,"   FREE";FR
400 PRINT #1,
410 PRINT #1,"FILE.NAME";TAB(11);"ATTR";TAB(20);"ADDRESS";
         TAB(32);"LOCATION"
420 PRINT #1,STRING$(45,"-"):CONSOLE 8,17
430 FTEND=DSKF(DN,10)-1
440 FOR S=1 TO FTEND
450  IF MAX=76 THEN DM$=DSKI$(DN,0,35,S)
460  IF MAX=39 THEN DM$=DSKI$(DN,1,18,S)
470  IF MAX=79 THEN DM$=DSKI$(DN,0,40,S)
480  IF MAX=34 THEN DM$=DSKI$(DN,18,S)
490  FOR I=0 TO 15
500    P=ASC(FI$(I))
510    IF P=0 THEN *NFILE
520    IF P=255 THEN *FEND
530    AT=ASC(AT$(I))
540    IF AT=0 THEN AT$="ASC":FT$=" "
550    IF AT=1 THEN AT$="MAC":FT$="*"
560    IF AT=&H10 THEN AT$="ASC+WTP":FT$=" "
570    IF AT=&H40 THEN AT$="ASC+RAW":FT$=" "
580    IF AT=&H80 THEN AT$="BAS":FT$="."
590    IF AT=&H90 THEN AT$="BAS+WTP":FT$="."
600    IF AT=&HA0 THEN AT$="BAS+PRT":FT$="."
610    IF AT=&HC0 THEN AT$="BAS+RAW":FT$="."
620    AD$=STRING$(9,".")
630    IF FT$="*" THEN GOSUB *ADR
640    PRINT #1,FI$(I);FT$;EX$(I);TAB(11);AT$;
           TAB(20);AD$;TAB(31);
650    CL=ASC(CL$(I))
660    WHILE CL<&HC0
670     PRINT #1," ";RIGHT$("0"+HEX$(CL),2);
680     CL=FA(CL)
690    WEND
700    PRINT #1,"(";MID$(STR$(CL-&HC0),2);")"
710   *NFILE
720  NN=NN+1:IF CP$="p" OR CP$="P" THEN 740
730  IF NN=15 THEN NN=0:PRINT:
      INPUT "Press RETURN ",DMY$:PRINT
740  NEXT I
750 NEXT S
760 '
770 *FEND
780 CONSOLE 0,25,1,0:LOCATE 0,24:PRINT:END
790 '
800 *ADR
810 F2$=DN$+":"+FI$(I)+EX$(I)
820 OPEN F2$ FOR INPUT AS #2
830 AD1$=RIGHT$("000"+HEX$(CVI(INPUT$(2,2))),4)
835 TMP!=CVI(INPUT$(2,2))
836 IF TMP!<0 THEN TMP!=TMP!+65536!
840 AD2$=RIGHT$("000"+HEX$(TMP!-1),4)
850 AD$=AD1$+"-"+AD2$
860 CLOSE #2 : RETURN
```

出力例

```
====== DRIVE  3  =======    FREE 138

FILE.NAME   ATTR       ADDRESS      LOCATION
------------------------------------------------
ABCD1       ASC        .........    49 48(2)
test2 *     MAC        0000-00FF    4C(2)
test3       ASC+WTP    .........    4D 4E(2)
test4       ASC+RAW    .........    47 46(2)
test5 .     BAS+WTP    .........    4F 50(1)
test7 .     BAS+PRT    .........    45 44(1)
test8 .     BAS+RAW    .........    51 52(1)
test6 .     BAS+WTP    .........    43 42(1)
Mac   *dat  MAC        0100-18FF    53 54 55 56(1)
```

7-5-2 1ファイル転送

N88-Disk BASIC(86)のシステムディスケットに入っているユーティリティプログラムの1つに〝xfiles.n 88″というのがあります。これは，8インチ・5インチ両面・片面のメディア間でプログラムやデータの転送ができるものです。しかしこれでは1枚のディスケットに入っているすべてのものが，転送されてしまいます。そこで次に紹介するプログラムは，任意のプログラムやデータを1個ずつ転送します。

ファイル転送プログラム

```
1 'save "Tfiles.n88
1000 ' ==== Tfiles.n88 ====
1010 '
1020 ' Transfer One File
1030 '
1040 ' Initialize
1050 DEFINT A-Z
1060 WIDTH 80,25
1070 D=15:DIM F$(D),E$(D),A$(D)
1080 FOR I=0 TO 15
1090  FIELD #0,I*16 AS DM$,6 AS F$(I),3 AS E$(I),1 AS A$(I)
1100 NEXT I
1110 FIELD #1,128 AS X1$,128 AS Y1$
1120 FIELD #2,128 AS X2$,128 AS Y2$
1130 '
1140 *START
1150 PRINT "■■■■ Transfer One File ■■■■" : PRINT
1160 INPUT "From Drive No. ";FD$ : FD=VAL(FD$)
1170 INPUT "To   Drive No. ";TD$ : TD=VAL(TD$)
1180  IF FD=0 AND TD=0 THEN *TRANSEND
1190 PRINT:INPUT "Ok (y or n) ";OK$
1200 IF OK$="y" OR OK$="Y" THEN 1220 ELSE CLS : GOTO *START
1210 '
1220 CLS
1230 PRINT "=== Drive ";FD;" ===   Free ";DSKF(FD):FILES FD :PRINT
1240 PRINT "=== Drive ";TD;" ===   Free ";DSKF(TD):FILES TD
1250 '
```

```
1260 PRINT:INPUT "Enter File Name to Transfer ";F$
1270  GOSUB *FILNAME
1280  DR=FD : GOSUB *SERCH :IF ER=1 THEN ER=0 : GOTO *START
1290    A$=A$(I) ' Attribute Set
1300  GOSUB *TRANS
1310  DR=TD:GOSUB *SERCH
1320  GOSUB *SETAT
1330 '
1340 PRINT FD;"--->";TD;F$
1350 FOR TM=1 TO 5000 : NEXT TM
1360 CLS : FILES FD : PRINT : FILES TD : GOTO *START
1370 *TRANSEND
1380 PRINT "Completed"
1390 END
1400 '
1410 *FILNAME
1420 P=INSTR(F$,"."):IF P<>0 THEN 1440
1430 F1$=LEFT$(F$+SPACE$(6),6):F2$=MID$(F$+SPACE$(9),7,3):GOTO 1460
1440 F1$=LEFT$(F$+SPACE$(6),P-1):F2$=MID$(F$+SPACE$(3),P+1,3)
1450 F1$=LEFT$(F1$+SPACE$(6),6):F2$=LEFT$(F2$+SPACE$(3),3)
1460 RETURN
1470 '
1480 '===== File Search =====
1490 *SERCH
1500 MAX=DSKF(DR,0)
1510 IF MAX=76 THEN SE=22 ELSE SE=12
1520  FOR S=1 TO SE
1530   IF MAX=76 THEN D$=DSKI$(DR,0,35,S)
1540   IF MAX=39 THEN D$=DSKI$(DR,1,18,S)
1550   IF MAX=34 THEN D$=DSKI$(DR,18,S)
1560   IF MAX=79 THEN D$=DSKI$(DR,0,40,S)
1570   FOR I=0 TO 15
1580    IF F1$=F$(I) AND F2$=E$(I) THEN RETURN
1590   NEXT I
1600  NEXT S
1610  BEEP:PRINT "*** File Not Found ***"
1620  FOR TM=1 TO 2000 : NEXT TM : ER=1 : RETURN
1630 '
1640 '===== File Transfer =====
1650 *TRANS
1660 OPEN FD$+":"+F$ AS #1
1670 OPEN TD$+":"+F$ AS #2
1680 FOR R=1 TO LOF(1):GET #1,R
1690 LSET X2$=X1$ : LSET Y2$=Y1$
1700 PUT #2,R :NEXT R :CLOSE #1,2
1710 RETURN
1720 '
1730 ' ===== Set Attribute =====
1740 *SETAT
1750  LSET A$(I)=A$
1760   IF MAX=76 THEN DSKO$ TD,0,35,S
1770   IF MAX=39 THEN DSKO$ TD,1,18,S
1780   IF MAX=34 THEN DSKO$ TD,18,S
1790   IF MAX=79 THEN DSKO$ TD,0,40,S
1800 RETURN
1810 '--------------------------
1820 ' === END OF SUBROUTINE ===
```

7-5-3 ファイルネーム・ソート

ディスケット内のファイルの数が多くなると目的のファイルを探すのが大変です。特に8インチの場合は手間がかかりますが，次のプログラムでファイル名をソートしておけばすぐ見つけ出せます。

```
1 'save "DIR.SRT"
100 DEFINT A-Z :DIM DDIR$(160),DIR$(15)
110 WIDTH 80,25:CONSOLE 0,25:CLS
120 PRINT "***** DIRECTORY SORT *****"
130 PRINT
140 INPUT "  Target Drive No. = ",TD
150 PRINT
160 INPUT "     Sure ? (y/n) ",A$:PRINT
170 IF A$<>"Y" AND A$<>"y" THEN BEEP :END
180 '
190 '*INIT
200  FOR I=0 TO 15
210    FIELD #0,16*I AS DUMMY$,16 AS DIR$(I)
220  NEXT I
230  MAXTRK=DSKF(TD,0) :DIRTRK=DSKF(TD,5)
240  MAXDIR=DSKF(TD,4)-2 :K=-1
250  IF MAXTRK=76 THEN MAXDIR=MAXDIR-1
260 '
270 '*GET.DIRECTORY
280  FOR I=0 TO DSKF(TD,1)-5
290    IF MAXTRK=76 THEN D$=DSKI$(TD,0,DIRTRK,I+1)
300    IF MAXTRK=39 THEN D$=DSKI$(TD,1,DIRTRK,I+1)
310    IF MAXTRK=34 THEN D$=DSKI$(TD,  DIRTRK,I+1)
320    IF MAXTRK=79 THEN D$=DSKI$(TD,0,DIRTRK.I+1)
330    FOR J=0 TO 15
340      IF ASC(DIR$(J))=255 THEN 400
350      IF ASC(DIR$(J))=0   THEN 380
360      K=K+1
370      DDIR$(K)=DIR$(J)
380    NEXT J
390  NEXT I
400 '
410   CONSOLE 7,25
420  PRINT STRING$(20,"-");CHR$(10);"Now sorting ..."
430 '*SORT.DIRECTORY
440  FOR I=K-1 TO 0 STEP -1
450   FOR J=0 TO I
460    IF DDIR$(J)>DDIR$(J+1) THEN SWAP DDIR$(J),DDIR$(J+1)
470   NEXT  J
480  NEXT I
490 '
500 C=0:FOR I=0 TO K
510  PRINT LEFT$(DDIR$(I),9)
520  C=C+1:IF C=14 THEN C=0:PRINT:
     INPUT "Press RETURN ",A$:PRINT
530 NEXT I
540 '
550 PRINT:PRINT:INPUT "Rewrite Directory (y/n) ";A$
560 IF A$<>"Y" AND A$<>"y" THEN *PEND
570 '
580 '*PUT.DIRECTORY
```

```
590 J=0 : SEC=1 : FOR I=0 TO K : LSET DIR$(J)=DDIR$(I)
600 J=J+1 : IF J=16 THEN GOSUB *DISKWT : J=0 : SEC=SEC+1
610 NEXT I : IF J=0 THEN *WTFF   ' -- Write FFH --
620 WHILE J<16 : LSET DIR$(J)=STRING$(16,255) : J=J+1 : WEND
630 GOSUB *DISKWT                ' --- DISK Write ---
640 *WTFF : IF J<>0 THEN SEC=SEC+1
645 IF MAXTRK=76 THEN MAXSEC=22 ELSE MAXSEC=12
650 FOR I=SEC TO MAXSEC
660   FOR J=0 TO 15 : LSET DIR$(J)=STRING$(16,255) : NEXT J
670   SEC=I : GOSUB *DISKWT      ' --- DISK Write ---
680 NEXT I : WIDTH 80,25 : PRINT "Completed." : FILES TD
690 *PEND : CONSOLE 0,25 : END ' --- Program END ---
700 *DISKWT                    ' --- Directory Write ---
710    IF MAXTRK=76 THEN DSKO$ TD,0,DIRTRK,SEC
720    IF MAXTRK=79 THEN DSKO$ TD,0,DIRTRK,SEC
730    IF MAXTRK=39 THEN DSKO$ TD,1,DIRTRK,SEC
740    IF MAXTRK=34 THEN DSKO$ TD,  DIRTRK,SEC
750 RETURN
```

7-5-4 オールマイティ・ディスク・ダンプ

次のプログラムは，ディスクの任意のセクタかファイル名を指定すると，その内容をダンプするものです。出力は1セクタごとにCRTとPRINTERの切り換えが可能です。

```
1 'save "D-dump"
100 '---- Almighty Disk Dump ----
110  ON KEY GOSUB 900,910,920,930,940,950,960,970,
        980,990
120  ON ERROR GOTO 1000 : ON STOP GOSUB 1020
130  KEY 1,"Restart":KEY 2,"Printer":KEY 3,"CRT"
140  KEY ON :STOP ON
150  WIDTH 80,25:CONSOLE 0,25:DEFINT A-Z
160  FIELD #0,128 AS A$(0),128 AS A$(1)
170  FIELD #1,128 AS A$(2),128 AS A$(3)
180  PRINT "***** Almighty Disk Dump Utility *****"
190  PRINT "    FUNCTION : F(ile or V(olume
200  PRINT "      f・1  key : Restart
210  PRINT "      f・2  key : Priter Output from Current Sector
220  PRINT "      f・3  Key : CRT Output from Next Sector
230  PRINT "      STOP key : Cancel this Utility
240  PRINT "      When CRT Output, Press any to continue."
250  PRINT ""
260  INPUT "FUNCTION        ";F$
270  IF F$="" THEN 1020 ELSE F$=LEFT$(F$,1)
280  IF F$="f" OR F$="F" THEN F$="F" :GOTO 320
290  IF F$="v" OR F$="V" THEN F$="V" :GOTO 510
300  GOTO 260
310 '---- FILE DUMP ROUTINE ----
320  INPUT "FILE NAME       ";FIL$
330  IF FIL$="" OR PF>0 THEN 1050 ' -------------  RESTART
340  OPEN FIL$ FOR INPUT AS #1 :CLOSE #1 :OPEN FIL$ AS #1
350  PRINT "SECTOR COUNT    ";LOF(1)
360  INPUT "START SECTOR    ";STARTSECTOR
370  IF STARTSECTOR>LOF(1) THEN 360
380  R=1 :IF STARTSECTOR<>0 THEN R=STARTSECTOR
390  IF R>LOF(1) THEN 490
400   L=2
```

```
410   GET #1,R
420   GOSUB 690
430   WHILE L$<>"L" AND B$="" AND PF=0:B$=INKEY$:WEND:B$=""
440   IF PF=1 THEN PF=0:CLOSE #1:GOTO 320
450   IF PF=2 THEN PF=0:L$="L":GOTO 420
460   IF PF=3 THEN PF=0:L$=""
470   R=R+1
480  GOTO 390
490  CLOSE #1 :GOTO 320
500 '---- VOLUME DUMP ROUTINE ----
510  INPUT "(DR,SU,TR,SE) ";DR,SU,TR,SE
520  IF DR<0 OR DR>8 THEN BEEP :GOTO 510
530  IF DR=0 OR PF>0 THEN 1050 ' ---------------- RESTART
540  IF DSKF(DR,2)=0 THEN IF SU<0 OR SU>1 THEN BEEP:GOTO 510
550  IF SU<0 OR SU>1 THEN BEEP :GOTO 510
560  IF TR<0 OR TR>DSKF(DR,0) THEN BEEP :GOTO 510
570  IF SE<1 OR SE>DSKF(DR,1) THEN BEEP :GOTO 510
580  IF DSKF(DR,2)=0 THEN DUM$=DSKI$(DR,TR,SE)
                     ELSE DUM$=DSKI$(DR,SU,TR,SE)
590   L=0
600   GOSUB 690
610   WHILE L$<>"L" AND B$="" AND PF=0:B$=INKEY$:WEND:B$=""
620   IF PF=1 THEN PF=0:GOTO 510
630   IF PF=2 THEN PF=0:L$="L":GOTO 590
640   IF PF=3 THEN PF=0:L$=""
650   SE=SE+1:IF SE=<DSKF(DR,1) THEN 580 ELSE SE=1
660   IF DSKF(DR,2)=1 THEN SU=SU+1:IF SU=1 THEN 580 ELSE SU=0
670   TR=TR+1:IF TR=<DSKF(DR,0) THEN 580 ELSE 510
680 '---- DUMP SUBROUTINE ------------------------------------
690  IF L$="L" THEN OPEN "lpt1:" FOR OUTPUT AS #2
               ELSE OPEN "scrn:" FOR OUTPUT AS #2
700  IF F$="F" THEN PRINT #2,"FILE NAME = ";FIL$;"
               RECORD NO = ";R
720  IF F$="V" THEN PRINT #2,"(DR,SU,TR,SE)=";DR;SU;TR;SE
730  PRINT #2,"     *0 *1 *2 *3 *4 *5 *6 *7 *8 *9 ";
740  PRINT #2,"*A *B *C *D *E *F     ";
750  PRINT #2," 0123456789ABCDEF"
760  FOR I=L TO L+1
770   FOR J=0 TO 7
780    I$=RIGHT$("00"+HEX$(128*I+J*16),2) :H$="" :C$=""
790    FOR K=1 TO 16
800     D$=HEX$(ASC(MID$(A$(I),J*16+K,1)))
810     H$=H$+RIGHT$("0"+D$,2)+" "
820     D$=MID$(A$(I),J*16+K,1):IF ASC(D$)<32 OR
        ASC(D$)>247 THEN D$="."
830     C$=C$+D$
840    NEXT K :PRINT #2,I$;"  ";H$;"    ";C$;:PRINT #2,""
850   NEXT J
860  NEXT I :ZZ=FRE(0)
870  CLOSE #2
880  RETURN
890 '---- FUNCTION KEY PROCESS ROUTINE ----
900 PF=1   :WHILE B$<>"":B$=INKEY$ :WEND :RETURN
910 PF=2   :WHILE B$<>"":B$=INKEY$ :WEND :RETURN
920 PF=3   :WHILE B$<>"":B$=INKEY$ :WEND :RETURN
930 PF=0   :WHILE B$<>"":B$=INKEY$ :WEND :RETURN
940 PF=0   :WHILE B$<>"":B$=INKEY$ :WEND :RETURN
950 PF=0   :WHILE B$<>"":B$=INKEY$ :WEND :RETURN
960 PF=0   :WHILE B$<>"":B$=INKEY$ :WEND :RETURN
970 PF=0   :WHILE B$<>"":B$=INKEY$ :WEND :RETURN
980 PF=0   :WHILE B$<>"":B$=INKEY$ :WEND :RETURN
```

```
990 PF=0  :WHILE B$<>"":B$=INKEY$ :WEND :RETURN
1000 PRINT "ERROR CODE = ";ERR
1010 IF ERR=53 THEN RESUME 320 ELSE 1020
1020 CLOSE :CLEAR
1030 KEY 1,"load "+CHR$(34):KEY 2,"auto ":KEY 3,"go to "
1040 END
1050 CLOSE :CLEAR :RUN
```

7-5-5 簡易ディスクエディター

このプログラムによりディスクの任意のトラックの１セクタが画面上でエディットできます。エディットは16進数で行い，そのチェックサムとアスキーダンプがリアルタイムに表示されます。

```
1 'save "D-edit.n88
100 ' ---- Disk Editor -----
110 '
120 DEFINT A-Z:WIDTH 80,25:CONSOLE 0,25,1,0
130 PRINT " ==== DISK EDITOR ===="
140  GOSUB *ADD      ' File Buffer Address
150 INPUT "Drive    No.    ";D:IF D=0 THEN *PEND
160 INPUT "Surface No.    ";SR
170 INPUT "Track    No.    ";T
180 INPUT "Sector   No.    ";S
190 PRINT CHR$(12);
200 PRINT " ==== DISK EDITOR ==== ";
210 PRINT USING " Drive #  Surface #  Track ##  Sector ##"
        ;D,SR,T,S
220 PRINT "*0 *1 *2 *3 *4 *5 *6 *7 *8 *9 *A *B *C *D *E *F";
230 PRINT "   SUM      ASCII DUMP";
240 GOSUB *DSKCHK
250 GOSUB *DSKRD
260 I=AD : PRINT
270 SUM=0:FOR J=0 TO 15
275 K=PEEK(I+J):PRINT RIGHT$("0"+HEX$(K),2);" ";
280 SUM=SUM+K:NEXT J:PRINT " ";RIGHT$("00"+HEX$(SUM),3);"  ";
290 FOR J=0 TO 15:K=PEEK(I+J)
300 IF K<32 OR K>247 THEN PRINT "."; ELSE PRINT CHR$(K);
310 NEXT J:PRINT:I=I+16:IF I-AD<256 THEN 270
320 ' ---------  Edit  ---------
330 CR$=CHR$(13):RT$=CHR$(28):LT$=CHR$(29)
335 UP$=CHR$(30):DN$=CHR$(31)
340 N=0:M=0
350 X=(N MOD 16)*3:Y=N ¥ 16+2
360 LOCATE X,Y:K$=INPUT$(1):IF M=1 THEN 420
370 IF K$=CR$ THEN *DEDIT
380 IF K$=RT$ THEN N=N+ 1:IF N>255 THEN N=0
390 IF K$=LT$ THEN N=N- 1:IF N<0   THEN N=255
400 IF K$=UP$ THEN N=N-16:IF N<0   THEN N=N+256
410 IF K$=DN$ THEN N=N+16:IF N>255 THEN N=N-256
420 IF K$>="0" AND K$=<"9" THEN 450
430 IF K$>="a" AND K$=<"f" THEN K$=CHR$(ASC(K$)-32):GOTO 450
440 ON M+1 GOTO 350,360
450 LOCATE X,Y:PRINT K$:IF M=0 THEN D$=K$:X=X+1:M=1:GOTO 360
460 M=0:D$=D$+K$:K=VAL("&H"+D$):POKE AD+N,K
```

```
470 SUM=0:FOR I=0 TO 15:SUM=SUM+PEEK(AD+(N ¥ 16)*16+I):NEXT
480 LOCATE 49,Y:PRINT RIGHT$("00"+HEX$(SUM),3)
490 LOCATE 54+N MOD 16,Y:IF K<32 OR K>247 THEN PRINT "・"
    ELSE PRINT CHR$(K)
500 N=N+1:IF N>255 THEN 330 ELSE 350
510 *DEDIT
520 LOCATE 0,19:INPUT "Edit OK (y/n) ";OK$
530  IF OK$="y" OR OK$="Y" THEN GOSUB *DSKWT
540 GOTO 150
550 ' ---- Program End -----
560 *PEND
570 END
580 ' ■■■■ Subroutines ■■■■■
590 ' ----- DISK READ -------
600 *DSKRD
610 IF TP=1 OR TP=2 OR TP=4 THEN DM$=DSKI$(D,SR,T,S)
620 IF TP=3 THEN DM$=DSKI$(D,T,S)
630 RETURN
640 '---- DISK WRITE -----
650 *DSKWT
660 IF TP=1 OR TP=2 OR TP=4 THEN DSKO$ D,SR,T,S
670 IF TP=3 THEN DSKO$ D,T,S
680 RETURN
690 ' ----- DISK TYPE CHECK -----
700 *DSKCHK
710 MX=DSKF(D,0)   ' Max Track
720 IF MX=76 THEN TP=1 ' 8 inch 2D
730 IF MX=39 THEN TP=2 ' 5 inch 2D
740 IF MX=34 THEN TP=3 ' 5 inch 1D
750 IF MX=79 THEN TP=4 ' 5 inch 2DD
760 RETURN
770 '
780 *ADD
790 AD=VARPTR(#0)+&H20 : SG=VARPTR(#0,1) :DEF SEG=SG
800 AD=PEEK(AD)+PEEK(AD+1)*256
810 RETURN
```

7-5-6 8インチIDリーダー

8インチのディスケットの0～76シリンダの合計154本のトラックについて，それぞれ1セクタのIDをリードする機能と特定のトラックの全セクタのIDを順次リードする機能があります。

前者は，ディスク全体のフォーマット分析，後者は1トラック内での論理セクタ番号の付けられ方を詳細に解析する場合に使います。なお，ディスケットはドライブ2に入れてプログラムを実行して下さい。

```
1 ' save "READ8.ID"
100 SCREEN 0,0 :CONSOLE 0,24,1,1 :CLEAR ,&H1A00 :DEFINT A-Z
110 PRINT "***** 8 inch FD analyze *****
120 PRINT
130 DEF SEG=&H1A00 :FOR I=0 TO &H78
140   READ A$ :POKE I,VAL("&H"+A$) :NEXT
150 DEF SEG=&H1B00 :FOR I=0 TO &H79
160   READ A$ :POKE I,VAL("&H"+A$) :NEXT
```

```
170 PRINT "Print device "
180 PRINT
190 INPUT "CRT or Printer  (c/p) ? ",D$
200 PRINT
210 PRINT "Function "
220 PRINT "t or T : Read ID from CC=0 to CC=76"
230 PRINT "s or S : Read ID 26 times on specified CCH"
240 PRINT "1      : Set  single density mode"
250 PRINT
260 INPUT "Select Function (t/s) ? ",F$
270 PRINT :  IF F$="1" THEN GOSUB 590 :GOTO 260
280 IF F$="t" OR F$="T" THEN *TRACK
290 IF F$="s" OR F$="S" THEN *SECTOR
300 PRINT "Invalid Function Specified ":BEEP :GOTO 260
310 *TRACK
320  DEF SEG=&H1A00 :CALL A
330  E=76 :GOSUB 430
340  END
350 *SECTOR
360  PRINT "Enter Cylinder and Head Address. To end,enter 99."
370  INPUT "  CC = ",CC
380  IF CC<0 OR CC>76 THEN IF CC=99 THEN END ELSE 370
390  INPUT "  H  = ",H  :IF H<>0 AND H<>1 THEN 390
400  DEF SEG=&H1B00 :CALL A(CC,H)
410  E=26/2-1 :GOSUB 430
420  PRINT :GOTO 360
430 '---- DUMP SUBROUTINE -------------------
440  IF D$="p" OR D$="P" THEN FF$="LPT1:" ELSE FF$="SCRN:"
450  OPEN FF$ FOR OUTPUT AS #2
460  PRINT #2,""
470  PRINT #2,"    AL AH BL BH CL CH DL DH ";
480  PRINT #2,"AL AH BL BH CL CH DL DH     "
490  PRINT #2,"                         CC SL RR  H ";
500  PRINT #2,"                 CC SL RR  H"
510  FOR J=&H10 TO E+&H10
520   H$=""
530   FOR K=0 TO 15
540    H$=H$+RIGHT$("0"+HEX$(PEEK(J*16+K)),2)+" "
550   NEXT K :PRINT #2,"    ";H$;"     "
560  NEXT J
570  CLOSE #2
580  RETURN
590 '---- PATCH FOR SINGLE SIDE VOLUME -----------
600  DEF SEG=&H1A00
610  POKE &H6,&H1A :POKE &H58,&H1A :POKE &H63,&H10
620  FOR I=&H64 TO &H6B :POKE I,&H90 :NEXT
630  RETURN
640 '---- MACHINE CODE -----
650 DATA 8C,C8,8E,D8,B8,91,5A,33,DB,33,C9,33,D2,89,87,00
660 DATA 01,89,9F,02,01,89,8F,04,01,89,97,06,01,CD,1B,73
670 DATA 36,F6,C4,E0,75,1B,F6,C4,C0,75,16,F6,C4,60,75,00
680 DATA 89,87,00,01,89,9F,02,01,89,8F,04,01,89,97,06,01
690 DATA CF,86,A7,01,01,87,8F,04,01,87,97,06,01,80,FC,1A
700 DATA 74,DE,B4,1A,E9,B6,FF,B4,5A,87,8F,04,01,87,97,06
710 DATA 01,83,C3,08,80,C6,01,80,FE,01,74,A1,32,F6,80,C1
720 DATA 01,80,F9,4D,7D,BA,E9,94,FF
730 DATA C4,77,04,26,8A,0C,C4,37,26,8A,34,8C,C8,8E,D8,B8
740 DATA 91,5A,33,DB,32,ED,32,D2,89,87,00,01,89,9F,02,01
750 DATA 89,8F,04,01,89,97,06,01,CD,1B,73,36,F6,C4,E0,75
760 DATA 1B,F6,C4,C0,75,16,F6,C4,60,75,00,89,87,00,01,89
```

```
770 DATA 9F,02,01,89,8F,04,01,89,97,06,01,CF,86,A7,01,01
780 DATA 87,8F,04,01,87,97,06,01,80,FC,1A,74,DE,B4,1A,E9
790 DATA B6,FF,8A,A7,01,01,87,8F,04,01,87,97,06,01,83,C3
800 DATA 08,81,FB,D0,00,7D,C4,E9,9E,FF
```

7-5-7 インテル HEX ファイル・ローダー

市販のソフトに CP/M-86 で作った HEX ファイルを Disk BASIC のファイルに転送するものがあります。しかし転送しただけでは，メモリーに BLOAD することはできません。これはファイルフォーマットが異なっているためです。次のプログラムは，インテル HEX ファイルを読み込んでメモリにロードし，最後に BSAVE を行うものです。なお，このローダーでは単一セグメントのプログラムを前提としています。

```
1 'save "LOAD.HEX"
100 CLEAR,&H1800
110 DEFINT A-Z:DIM C(3)
120 INPUT "Intel HEX Format File name ";F$
130 OPEN F$ FOR INPUT AS #1
140 '
150 *INPUT.HEX.LECORD
160 LINE INPUT #1,I.HEX.L$
170 C(0)=VAL("&H"+MID$(I.HEX.L$,2,2))
180 C(1)=VAL("&H"+MID$(I.HEX.L$,4,4))
190 C(2)=VAL("&H"+MID$(I.HEX.L$,8,2))
200 '
210 IF (C(2)>&H80) AND (C(2)<&H85) THEN *DATA.POKE
220 IF C(2)=0    THEN *DATA.POKE
230 IF C(2)=1    THEN *END.OF.FILE
240 IF C(2)=3    THEN *STRT.AD.MK
250 IF C(2)=&H85 THEN
      PRINT "absolute code   segment":GOTO *SET.SEG
260 IF C(2)=&H86 THEN
      PRINT "absolute data   segment":GOTO *SET.SEG
270 IF C(2)=&H87 THEN
      PRINT "expand   stack  segment":GOTO *SET.SEG
280 IF C(2)=&H88 THEN
      PRINT "absolute expand segment":GOTO *SET.SEG
290 GOTO *INPUT.HEX.LECORD
300 '
310 *SET.SEG
320   DEF SEG=VAL("&H"+MID$(I.HEX.L$,10,4))
330 '
340 *STRT.AD.MK
350   LD.SEGMENT=VAL("&H"+MID$(I.HEX.L$,10,4))
360   ST.ADDRESS=VAL("&H"+MID$(I.HEX.L$,14,4))
370 GOTO *INPUT.HEX.LECORD
380 '
390 *DATA.POKE
400   DTLEN=C(0):ADD=C(1)
410   FOR I=0 TO DTLEN-1
420     PK.ADDRESS=ADD+I
430     POKE PK.ADDRESS,VAL("&H"+MID$(I.HEX.L$,2*I+10,2))
440   NEXT
```

```
450 GOTO *INPUT.HEX.LECORD
460 '
470 *END.OF.FILE
480   CLOSE 1:PRINT "LOAD END !"
490   PRINT "SEGMENT address:";
             RIGHT$(("0000"+HEX$(LD.SEGMENT)+"H"),5)
500   PRINT "START   address:";
             RIGHT$(("0000"+HEX$(ST.ADDRESS)+"H"),5)
510   PRINT "END     address:";
             RIGHT$(("0000"+HEX$(PK.ADDRESS)+"H"),5)
520 INPUT "Saving object code File name ";FS$
530 BSAVE FS$,ST.ADDRESS,PK.ADDRESS+1
540 END
```

第 8 章　プリンタ出力

8-1　画面コピー機能

8-2　テキスト画面のコピー

8-2-1　BASICによるサブルーチン
8-2-2　マシン語によるサブルーチン

8-3　カラーグラフィックコピー

8-3-1　640×200モード
8-3-2　640×400モード

8-4　アセンブリ言語によるプリンタ出力

8-4-1　イニシャライズ
8-4-2　1バイト出力
8-4-3　複数バイト出力

8-5　PRINT/LPRINTあれこれ

8-5-1　出力デバイス名の変更
8-5-2　切り換えルーチンを作る
8-5-3　内部ルーチンを利用する
8-5-4　未使用コマンドでの切り換え

第8章　プリンタ出力

8-1　画面コピー機能

N_{88}-BASIC(86)は，COPY文とCOPYキーにより，画面上に表示されている文字やグラフィックをプリンタに出力する機能を持っています。

テキスト画面（文字，漢字）およびグラフィック画面（640×200ドット，640×400ドット）に対応して，コピーの形式が5種類用意されています。COPYキーについては，CTRLキーおよびGRPHキーとの組み合わせにより3通りの方法があります。表8-1にその機能一覧をまとめてみます。

COPY　1	テキスト画面のみをプリンタにコピーします。	CTRL+COPYキー
COPY　2	グラフィック画面のみをプリンタにコピーします。	GRPH+COPYキー
COPY　3	テキスト画面，グラフィック画面の両方をプリンタにコピーします。	COPYキー
COPY　4	グラフィック画面のみをプリンタにコピーします（漢字出力用）。640×200ドットのモードで出力された漢字（グラフィックパターン）はこのモードで出力すると上下が2分の1に縮少されてコピーされます。	
COPY　5	テキスト画面，グラフィック画面の両方をプリンタにコピーします。4と同じく640×200のモードで出力されたパターンは2分の1に縮小されます。	

表8-1 コピー機能一覧

では次にCOPYの1～5までの機能を調べるプログラム例を紹介します。

画面コピーデモプログラム例1（640×200 モード）

```
1 'save "SC.dmo"
100 ' Screen Copy Demo 1 ...640 x 200
110 CONSOLE 0,25,0,0 : WIDTH 40,25 :SCREEN 0,0
120 CLS 3 : LOCATE 7,8:PRINT "Screen Copy Demo "
130  X=120:Y=100
140 FOR I=1 TO 9 :READ KC$ : KC=VAL("&H"+KC$)
150  PUT (X,Y),KANJI(KC)
160  X=X+20
170 NEXT I
180 LOCATE 7,10
190 PRINT "画面コピ-DEMO"
200 LOCATE 7,16
210 PRINT "ﾌﾟﾘﾝﾀ ｱｳﾄﾌﾟｯﾄ"
```

```
220 LINE (100,50)-(600,150),7,B
230 FOR I=1 TO 10
240 CIRCLE (500,100),I*I
250 NEXT I
260 FOR I=2 TO 8  STEP 2
270  PAINT (499-I*I, 99),5,7
280 NEXT I
290 FOR I=1 TO 5
300 LPRINT "COPY";I
310  COPY I
320 NEXT I
330 DATA 3268,4C4C,2533,2554,213C,2344,2345,234D,234F
340 END
```

画面コピーデモプログラム例２（640×460 モード）

```
1 'save "SC2.dmo"
100 ' Screen Copy Demo 2 ...640 x 400
110 CONSOLE 0,25,0,0 : WIDTH 40,25 :SCREEN 3,0
120 CLS 3 : LOCATE 7,8:PRINT "Screen Copy Demo "
130  X=120:Y=100
140 FOR I=1 TO 9 :READ KC$ : KC=VAL("&H"+KC$)
150  PUT (X,Y),KANJI(KC)
160  X=X+20
170 NEXT I
180 LOCATE 7,10
190 PRINT "画面コピーＤＥＭＯ"
200 LOCATE 7,16
210 PRINT "ﾌﾟﾘﾝﾀ ｱｳﾄﾌﾟｯﾄ"
220 LINE (100,50)-(600,300),7,B
230 FOR I=1 TO 10
240 CIRCLE (500,200),I*I
250 NEXT I
260 FOR I=2 TO 8  STEP 2
270  PAINT (499-I*I,198),5,7
280 NEXT I
290 FOR I=1 TO 5
300 LPRINT "COPY";I
310  COPY I
320 NEXT I
330 DATA 3268,4C4C,2533,2554,213C,2344,2345,234D,234F
340 END
```

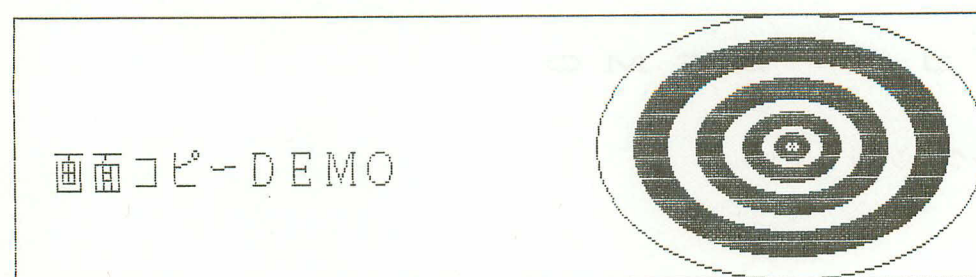
Screen Copy Demo

画面コピーＤＥＭＯ

ﾌﾟﾘﾝﾀ ｱｳﾄﾌﾟｯﾄ

COPY 1

COPY　2

COPY　3

COPY　4

COPY　5

8-2　テキスト画面のコピー

8-2-1 BASICによるサブルーチン

コピーキーでテキスト画面をとるとLPRINTで出力した文字と異なっています。この画面コピーサブルーチンは，LPRINTと同じ字体でプリンタに出力できます。さらに画面コピーをするスタートとエンドの行を指定することができます。ただし，これは80×25文字モードです。

テキスト画面コピープログラム1

```
1 'save"TXTCOP"
59999 *TXTCOPY
60000 DEF SEG=&HA000
60010  FOR I=S TO E
60020    FOR J=0 TO 159 STEP 2:P=PEEK(I*160+J)
60030  IF P>31 AND P<248 THEN LPRINT CHR$(P); ELSE LPRINT " ";
60040  NEXT J:LPRINT
60050 NEXT I
60060 RETURN
Ok

S=0:E=15:GOSUB *TXTCOPY
Ok
```

これはサブルーチン形式をとっており，Sにプリント開始行，Eに最終行を入れてGOSUB　*TXTCOPYとすれば，SとEで指定した範囲がプリントアウトされます。ただし，漢字は出力されません。漢字コードは無視していますので，もし漢字も出力したい場合はなんとか工夫してみて下さい。上記はLISTをとった後，テキスト画面コピールーチンをコールしたものです。

8-2-2 マシン語によるサブルーチン

前出のルーチンはBASICでも，スピードはそんなに遅くありませんが，どうしてもマシン語で出力したい方に次のルーチンを紹介します。

使い方は，S%に開始行，E%に最終行を入れて，DEF SEG=&H1F00：TXT=0：CALL TXT(S%，E%)として下さい。なお，アセンブラレベルでのプリンタ出力は後の節で説明しています。

テキスト画面コピープログラム2

```
1 'save "TXTCOP.MAC"
100 ' TEXT COPY MACHINE SUBROUTINE
110 ' ---  S%=START:E%=END:T=0:CALL T(S%,E%)
120 DEF SEG=&H1F00
130  FOR I=0 TO &H44 : READ D$
140    D=VAL("&H"+D$) : POKE I,D
150  NEXT I
160 END
170 DATA C4,37,26,8B,0C,C4,77,04,26,8B,1C,B8,00,A0,8E,D8
180 DATA 53,33,F6,8B,C3,BA,A0,00,F7,E2,8B,D8,8D,38,8A,05
190 DATA 3C,1F,77,02,7A,04,3C,F8,72,02,B0,20,E8,0F,00,46
200 DATA 46,81,FE,A0,00,75,E5,5B,43,3B,D9,75,D3,CF,56,B4
210 DATA 11,CD,1A,5E,C3,00,00,00,00,00,00,00,00,00,00,00
```

使い方

```
1 ' save "TXTCOP.BAS"
10 CLEAR ,&H1F00:DEF SEG=&H1F00
20 TXT=0
30 S%=0:E%=10
40 CALL TXT(S%,E%):LPRINT
50 END
```

8-3 カラーグラフィックコピー

8-3-1 640×200モード

前出のサンプル出力で分かるとおり，真円をコピーするとたて長になってしまいます。これはN_{88}-BASIC (86) のコピールーチンがなせるわざです。真円のコピーをとるためにはBASICかアセンブリ言語で，そのプログラムを作るしかありません。そこで次に紹介するのが1：1.03の比で真円がコピーできるプログラムです。さらにカラー対応となっています。

これは，カラーグラフィック画面をプリンタ用紙1枚分の大きさに引き伸ばしてコピーするもので，カラーに応じて4段階の濃淡が付きます。ただし，これは640×200ドットのモードで，画面全体のコピーには約3～4分かかります。なお，これはPC−8821/22用です。

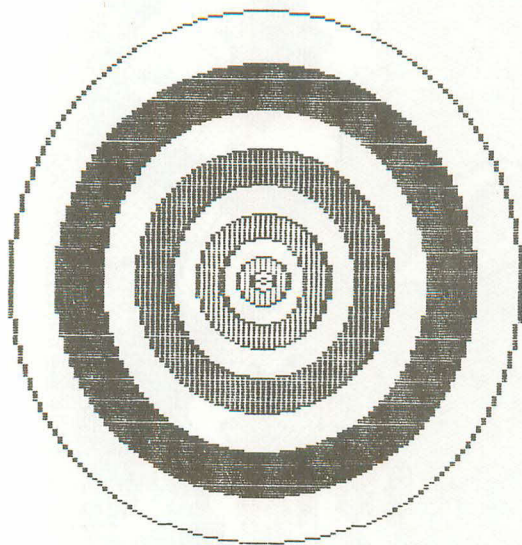
カラー対応画面コピー (640×200ドット)

```
1 'save "SC200.BAS"
100 ' Graphic Screen Copy 640 x 200
110 ' ---  SC=0:CALL SC ---
120 DEF SEG=&H1D00
130  FOR I=0 TO &HC1 : READ D$
140    D=VAL("&H"+D$) : POKE I,D
150  NEXT I
160 END
170 DATA 1E,0E,0E,1F,07,BB,B0,00,B9,0A,00,E8,8F,00,8C,16
180 DATA C6,00,89,26,C8,00,8C,D8,8E,D0,BC,4A,01,CD,A0,33
190 DATA D2,BB,BA,00,B9,06,00,E8,73,00,BB,C7,00,BE,60,01
200 DATA 33,C0,B9,02,00,89,04,46,46,E2,FA,BD,01,00,B9,08
210 DATA 00,33,FF,89,1E,C4,00,53,52,B8,08,00,F7,E2,8B,D8
220 DATA 8D,01,A3,C2,00,E8,4A,00,D0,F8,3C,03,74,10,3C,00
230 DATA 75,02,B0,03,BE,60,01,01,2C,46,FE,C8,75,F9,03,ED
240 DATA 47,E2,DD,BB,60,01,B9,04,00,E8,21,00,5A,5B,4B,83
250 DATA FB,FF,75,A9,BB,C0,00,B9,02,00,E8,10,00,42,83,FA
260 DATA 50,75,8E,8E,16,C6,00,8B,26,C8,00,1F,CF,B4,30,CD
270 DATA 1A,C3,53,51,57,55,BB,C2,00,CD,AF,5D,5F,59,5B,C3
280 DATA 1B,4D,1B,3E,1B,54,31,36,0D,0A,1B,53,30,38,30,30
290 DATA 0D,0A,00,00,00,00,00,00,00,00,00,00,00,00,00,00
```

使い方は，このプログラムを実行後，画面コピーをとりたいときにダイレクトで次のステートメントを実行するかプログラム中に入れておいて下さい。

DEF SEG=&H1D00：SC=0：CALL SC

なお、PC−8023(C)で出力するにはCALLする前にLPRINT CHR$(27);CHR$ (&H51); として下さい。

真円コピーサンプル

グラフィックスコピーサンプル（640×200ドット）システムソフト・オリジナル作品

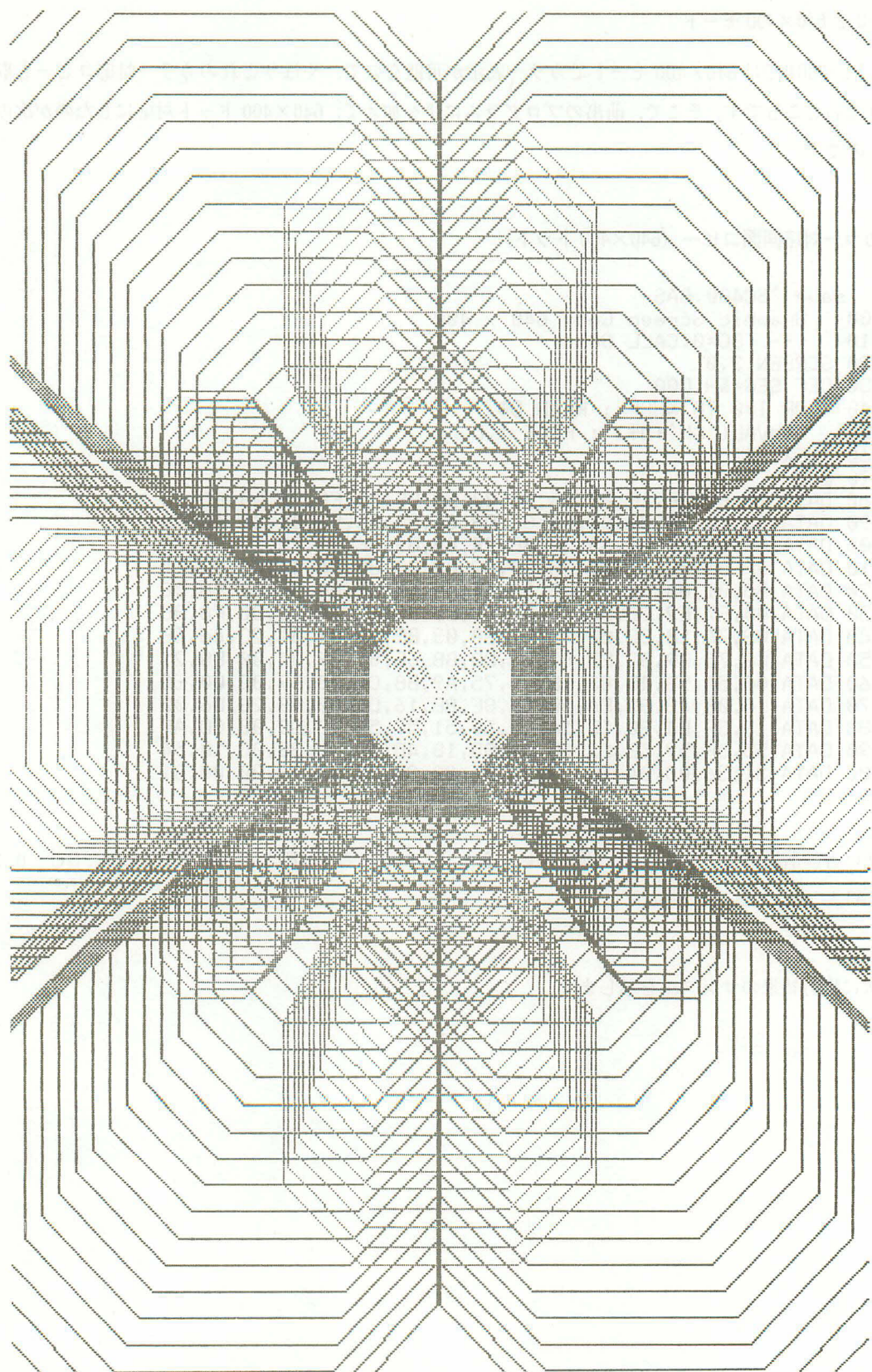
8-3-2 640×400 モード

PC-9801 では 640×400 モードでカラー表示が可能なので，やはりこれのカラー対応コピーも取りたいところです。そこで，前出のプログラムに手を加えて，640×400 ドット対応にしたのが次のものです。

カラー対応画面コピー（640×400 ドット）

```
1 'save "SC400.BAS
100 ' Graphic Screen Copy 640 x 400
110 ' ---  SC=0:CALL SC ---
120 SCREEN 3,0
130 DEF SEG=&H1D00
140  FOR I=0 TO &HCF : READ D$
150    D=VAL("&H"+D$) : POKE I,D
160  NEXT I
170 END
180 DATA 1E,0E,0E,1F,07,BB,B5,00,B9,0F,00,E8,94,00,8C,16
190 DATA D4,00,89,26,D6,00,8C,D8,8E,D0,BC,58,01,CD,A0,BB
200 DATA CC,00,CD,A1,33,D2,BB,C4,00,B9,06,00,E8,73,00,BB
210 DATA 8F,01,BE,6E,01,33,C0,B9,02,00,89,04,46,46,E2,FA
220 DATA BD,01,00,B9,08,00,33,FF,89,1E,D2,00,53,52,B8,08
230 DATA 00,F7,E2,8B,D8,8D,01,A3,D0,00,E8,4A,00,D0,F8,3C
240 DATA 03,74,10,3C,00,75,02,B0,03,BE,6E,01,01,2C,46,FE
250 DATA C8,75,F9,03,ED,47,E2,DD,BB,6E,01,B9,04,00,E8,21
260 DATA 00,5A,5B,4B,83,FB,FF,75,A9,BB,CA,00,B9,02,00,E8
270 DATA 10,00,42,83,FA,50,75,8E,8E,16,D4,00,8B,26,D6,00
280 DATA 1F,CF,B4,30,CD,1A,C3,53,51,57,55,BB,D0,00,CD,AF
290 DATA 5D,5F,59,5B,C3,1B,51,0D,1B,4D,0D,1B,3E,0D,1B,54
300 DATA 31,36,0D,0A,1B,53,31,36,30,30,0D,0A,03,00,00,01
```

使い方は前出のものと同じで，画面コピーをとりたいところでDEF SEG＝&H１D00：SC＝０：CALL SC として下さい。

次に出力結果のサンプルを示します。

グラフィックスコピーサンプル（640×400ドット）NEC PC-9801F のデモプログラムより

8-4 アセンブリ言語によるプリンタ出力

ここでは，アセンブラレベルでセントロニクス系プリンタに出力するルーチンを紹介します。ODA系のプリンタ出力についても同様で，両者の選択は，メモリスイッチSW 3の第5ビットによって行います。0のときセントロニクス系での1のときODA系となります。

8-4-1 イニシャライズ（初期化）

このイニシャライズは，インタラプトコール1AHで，セントロニクス系プリンタのインターフェイス μPD 8255 Aを初期化し，プリンタのステータス（送信の可，不可）情報を得ます。ステータスは，AHレジスタの最下位ビットにセットされ，1のときデータ送信可能，0のときデータ送信不可となります。次にプリンタがBUSYかREADYであるかの判断を行うプログラム例をあげます。

```
; PRINTER BUSY CHECK
;
BUSY:    MOV AH,10H          ; BIOS コマンド
         INT 1AH             ; イニシャライズ
         TEST AH,01H         ; インタラプトコール
         JNE BUSY
READY:
;
```

8-4-2 1バイト出力

プリンタがREADYの状態になったら，いよいよ出力です。次のようにレジスタをセットし，インタラプトコール（1 AH）をするとプリンタに1バイト出力します。

```
;
; LPRINT ONE BYTE
;
PUTCHR: MOV AL,41H          ; ("A" 出力するアスキーコード)
        MOV AH,11H          ; BIOS コマンドコード
        INT 1AH
        IRET
;
```

次にA～Zの26文字を出力する例をあげます。

```
;
; LPRINT A TO Z
;
START:  MOV CX,26       ; COUNTER
OUTAZ:  MOV AL,41H      ; AL='A'
        MOV AH,11H      ; BIOS CODE
OUTLP:  PUSH AX         ; SAVE AH & AL
        INT 1AH         ; INT CALL
        POP AX          ; RESTORE AH & AL
        INC AL          ; AL=NEXT CHR
        LOOP OUTLP
        MOV AL,0DH      ; CR
        MOV AH,11H
        INT 1AH
        IRET
```

8-4-3 複数バイト出力

アセンブラレベルでは1バイトずつ出力していたのではめんどうでバイト数も多くなります。そこで，複数のバイトを一度に出力する方法を紹介しましょう。レジスタを次のようにセットします。

```
;
; LPRINT A NUMBER OF BYTES
;
NBYTES:
        MOV AH,30H          ; BIOS CODE
        MOV BX,ES:OFFSET DATA
        MOV CX,15                       ; 出力するバイト数
        INT 1AH
;
DATA DB 'Sample Data ...'
```

出力するデータは，その格納先頭アドレスをESでセグメント，BXでオフセットを設定します。次にサンプルを示します。

```
                        ;************************
                        ;* LPRINT ROUTINE
                        ;************************
                        ;
0000 B430               LPRINT: MOV AH,30H        ; BIOS COMMAND
0002 BA001F                     MOV DX,1F00H      ;
0005 8EC2                       MOV ES,DX         ; LPRINT DATA SEGMENT
0007 BB0F00                     MOV BX,ES:OFFSET PDATA
000A B91D00                     MOV CX,29         ; NO.OF BYTES TO LPRINT
000D CD1A                       INT 1AH
                        ;
  000F                  DATA    EQU     OFFSET $
                                ESEG
                                ORG DATA
                        ;
000F 0D0A54686973       PDATA   DB 0DH,0AH,'This is a sample program.',0DH,0AH
     206973206120
     73616D706C65
     2070726F6772
     616D2E0D0A

                                END
```

```
This is a sample program.
```

8-5 PRINT／LPRINTあれこれ

画面表示とプリンタ出力を切り換えたり，同時に出力したりする方法をあれこれと考えてみたいと思います。

8-5-1 出力デバイス名の変更

一番簡単な方法は，ファイルディスクリプタを変数に入れて，必要に応じて出力先を変更することです。

```
1 ' save "PL1"
100 INPUT "CRT(c) or PRINTER (p) ";CP$
110 IF CP$="c" OR CP$="C" THEN F$="SCRN:"
120 IF CP$="p" OR CP$="P" THEN F$="LPT1:"
130 '
140 OPEN F$ FOR OUTPUT AS #1
150 FOR I=1 TO 3
160  READ D$:PRINT #1,D$
170 NEXT I
180 CLOSE #1
190 END
200 DATA This is a demo.
210 DATA CRT or PRINTER
220 DATA Device Change
```

8-5-2 切り換えルーチンを作る

BASICのPRINT／LPRINT のステートメントは使わず，それらに対応する独自のルーチンを作ったのが次のものです。

CALL文を用いてパラメータをセットすればOKです。ただし，これにはCR／LFは入っていません。

```
DEF  SEG=&H1F00:PL=0
CALL  PL (P%, D$)
          ↑         ↑
          ├0-CRT    └──出力データ
          └1-PRINTER
```

```
1 ' save "P/L.bas"
100 DEF SEG=&H1F00
110 PL=0:DIM D$(5)
120 FOR I=0 TO &H4C
130  READ D$ :  D=VAL("&H"+D$)
140  POKE I,D
150 NEXT I
160 FOR I=1 TO 5
170  READ D$(I)
180 NEXT I
190' PRINT
200  P%=0
210 GOSUB *PL
220 P%=1
230 GOSUB *PL
240 END
250 ' --- SUB ---
260 *PL
270 FOR I=1 TO 5
280  D$=D$(I)
290  D$=D$+CHR$(13)+CHR$(10)
300  CALL PL(P%,D$)
310 NEXT I
320 RETURN
330 DATA C4,37,26,8B,0C,80,FD,00,74,05,32,ED,BA,60,00,C4
340 DATA 7F,04,26,8B,05,26,8B,74,02,8E,DA,3D,00,00,74,06
350 DATA 3D,01,00,74,1A,CF,B8,60,00,8E,C0,51,BF,02,02,FC
360 DATA F3,A5,59,8E,D8,CD,C2,B8,00,A0,8E,C0,CD,89,CF,1E
370 DATA 07,56,5B,B8,60,00,8E,D8,B4,30,CD,1A,CF
380 ' OUTPUT DATA
390 DATA This is a test program.
400 DATA It is for PRINT/LPRINT.
410 DATA " ■■■ Chapter 8 ■■■"
420 DATA "    PRINTER OUTPUT"
430 DATA "  PC-TechKnow 9800"
```

8-5-3 内部ルーチンを利用する

内部ルーチンにカレントデバイスへの出力というのがあります。

これは，

```
MOV    CX，出力するバイト数
MOV    DI，25H
INT    C4
```

という形でコールします。

その前に出力先を決めるセグメント60Hのワークエリア(1840H)に値をセットしておく必要があります。

```
〔1840H〕 ←―― 3（プリンタ）
〔1840H〕 ←―― 4（CRT）
```

また，出力するバイト数は1884Hに入っています。これをCXに入れておきます。

```
MOV    CX，  〔1884H〕
```

なお，出力先をプリンタにした場合には，出力する文字数の何文字目でCR／LFを入れるかを決定するワークエリア（153AH）に値をセットしておかなければなりません（WIDTH LPRINTのパラメータ格納エリア)。

0，1 ………1文字ずつCR／LF
2～79……その数値に対応する文字ずつCR／LF
80…………1行80文字ごとにCR／LF

普通，プリンタは1行80字でCR／LFしますので，80文字に満たない場合は出力文字数を入れておくとよいでしょう。

使い方は，PRINT文実行直後にマシン語ルーチンをコールすると，それがプリンタにも出力されます。

```
PRINT "ABC"：CALL  A
```

また，1840Hに4を入れると，

```
LPRINT "ABC"：CALL  A
```

で，CRT にも同じものが出力されます。

```
1 ' save "PL2"
100 DEF SEG=&H1F00 : P=0
110 ' -- PRINT AND LPRINT ALSO --
120 FOR I=0 TO &H16 : READ D$
130  D=VAL("&H"+D$) : POKE I,D
140 NEXT I
150 FOR I=1 TO 3
160  READ D$:PRINT D$ : CALL P
170 NEXT I
180 END
190 '
200 DATA 16              ' PUSH SS
210 DATA 1F              ' POP DS
220 DATA B0,03           ' MOV AL,3
230 DATA BB,40,18        ' MOV BX,1840
240 DATA 88,07           ' MOV [BX],AL
250 DATA 8B,0E,84,18     ' MOV CX,[1884]
260 DATA 89,0E,3A,15     ' MOV [153A],CX
270 DATA BF,25,00        ' MOV DI,25
280 DATA CD,C4           ' INT C4
290 DATA CF              ' IRET
300 '
310 DATA This is a demo.
320 DATA CRT and PRINTER
330 DATA Interrupt Call
```

8-5-4 未使用コマンドでの切り換え

N_{88}-BASIC(86)の未使用コマンドであるCMDを用いてPRINTとLPRINTの切り換えを行ってみましょう。

コマンドは次のように割り当てます。

CMD　ON…………PRINT→LPRINT
CMD　OFF………上記を解除

なお，未使用コマンドの使い方は第14章ランダムテクニックを参照して下さい。

次のプログラムを実行させると，以後，CMD ON とCMD OFF のコマンドが使えるようになります。これらはダイレクトモードでもプログラム中でもOKです。LPRINT をPRINT の機能に変えるには，350行のデータを2バイト書き換えて下さい。

```
1 ' save "cmd.bas"
10 PRINT "CMD ON/OFF Sample"
20 ' print
30 PRINT "PC-TechKnow 9800"
40  CMD ON
```

```
50 PRINT "Printer Output"
60  CMD OFF
70 PRINT "CRT Output"
80 END
```

CMD ON・OFF による PRINT→LPRINT

```
1 'save "cmdp"
100  DEF SEG=&H1F00 : RESTORE 310
110  FOR I=0 TO &H5D  '--- Machine Code
120   READ D$:D=VAL("&H"+D$)
130   POKE AD+I,D
140  NEXT
150 DEF SEG=&H60 : RESTORE 290
160 FL=&H1593  ' -- CMD Command Flag
170 AD=&H159C
180 POKE FL,0
190 FOR I=0 TO 7 ' -- Set User Routine Address
200  READ D$:D=VAL("&H"+D$)
210  POKE AD+I,D
220 NEXT
230 POKE FL,1  ' -- Flag On
240 '---------------------------
250 PRINT "You can use the following commands:"
260 PRINT "  CMD ON : PRINT -> LPRINT
270 PRINT "  CMD OFF: Cancel
280 END
290 DATA 00,00,00,1F
300 DATA 3E,00,00,1F
310 DATA 3C,E8,F8,75,39,E8,0D,00,80,FB,BD,74,18,80,FB,BF
320 DATA 74,2C,E9,0A,00,89,36,EA,06,BF,0D,00,CD,C4,C3,BF
330 DATA 01,00,CD,C4,CB,BB,9C,15,B8,40,00,89,07,E9,07,00
340 DATA BB,9C,15,33,C0,89,07,A1,EA,06,A3,E8,06,F9,CB,90
350 DATA 3C,C0,74,07,3C,E8,74,07,E9,02,00,B0,AD,F8,CB,E8
360 DATA C3,FF,80,FB,BF,74,D9,80,FB,BD,74,F1,EB,C1,90,90
```

C0 → AD

AD → C0　2バイト書き換えると LPRINT → PRINT となります。

第９章　漢　　字

9-1　漢字ROMボード
9-2　テキスト画面とグラフィック画面に漢字表示
9-3　ファンクションキーエリアに漢字表示
9-4　漢字フォントパターン読み出し
9-5　漢字フォントパターンの拡大表示
9-6　漢字・JISコード対応表示
9-7　漢字フォントをビットイメージで出力
9-8　任意のフォントの作成・出力

第9章　漢字

9-1　漢字ROMボード

PC-9801にオプションの漢字ROMボードを装着すると，日本語表示が可能となります。さらに専用高解像度（640×400ドット）ディスプレイを使用すると，テキスト画面に日本語表示ができます。

PC-9801が扱う日本語文字はテキスト画面とグラフィック画面により表9-1のようになっています。

表9-1 使用できる日本語文字

9-2 テキスト画面とグラフィック画面に漢字表示

日本語文字は，1バイトの英数カナ文字とは異なり，1文字が2バイトのコードで表わされます。N_{88}-BASIC（86）では，日本語の文字列を一般の1バイトの文字列と異なったものとして扱うようなことはせず，両者を混在させて扱うことができます。

そこで，1バイトの文字列と漢字とを区別するために，漢字文字列の始めと終りにそれぞれ，漢字イン（KI＝1 B 4 BH），漢字アウト（KO＝1 B 48 H）の記号を付けています。試しに次のプログラムを実行して，KI，KOを調べてみましょう。

漢字イン・アウトチェック

```
1 'save "kanji"
10 ' KANJI SAMPLE
20 A$="KANJI 漢字 ｶﾝｼﾞ"
30 FOR I=1 TO LEN(A$)
40  B$=MID$(A$,I,1)
50  PRINT HEX$(ASC(B$));" ";
60 NEXT I
```

```
4B 41 4E 4A 49 20 1B 4B 34 41 3B 7A 1B 48 20 B6 DD BC DE
K  A  N  J  I  ␣  \___/ \___/ \___/ \___/  ␣  ｶ  ﾝ  ｼ  ﾞ
                   KI    漢    字    KO
```

テキスト画面への漢字表示は，CTRL ＋ XFER やKINPUTなどでJISコードを入力すれば簡単にできます。さらにBASICのプログラム中にPRINT文で出力したり，文字変数にも代入できます(前ページのサンプルプログラム参照)。

しかし，グラフィック画面に表示するには少しめんどうです。次のように表示するX，Y座標とJISコードを設定する必要があります。

PUT (X, Y), KANJI (JISコード)

次に紹介するプログラムは，表示するX，Y座標，縦表示・横表示のフラグZを指定するだけでグラフィック画面の任意の位置に漢字を表示するものです。

漢字のG-VRAM表示

```
1 'save "PkanjI"
100 X=10:Y=10:Z=1
110 FOR I=1 TO 8
120 READ KC$
130 KC=VAL("&H"+KC$)
140   PUT (X,Y),KANJI(KC)
150   GOSUB *LCT
160 NEXT
170 END
180 *LCT
190 IF Z=1 THEN X=X+20:GOTO 210
200 Y=Y+20
210 RETURN
220 DATA 3441,3B7A,493D,3C28
230 DATA 2344,2345,234D,234F
```

9-3 ファンクションキーエリアに漢字表示

漢字をファンクションキーに定義することができないのに，市販の漢字ワープロなどではちゃんとそのエリアに表示して，それに対応するキーが押されると，その処理を行うようになっています。

これは実は，ファンクションキーエリアの位置にPRINT文で漢字を書き，それをリバースさせて，あたかもファンクションキーであるかのように見せかけているだけです。そして，ファンクションキー割り込みを使って，それぞれに対応する処理を行うようにしています。

そこで以上のことをプログラミングした簡単なメニュー・ルーチンを紹介します。各ファンクションキーを押すと，それに対応したメッセージが表示されます。別な処理項目を選ぶときには RET

キーを押してメニューに戻して下さい。

漢字メニュールーチン

```
1 'save "Kanji.key"
1000 ' Display Kanji
1010 '  in Function Key Area
1020  CONSOLE 0,25,0,0 :DEFINT X,Y :CLS
1030 'FOR I=1 TO 10 : KEY I,"" : NEXT I ' -- Key Clear
1040 RESTORE 1500 : READ A$
1050  X=4:Y=24:GOSUB *DISP ' -- [f·1]
1060 FOR I=1 TO 4 : READ A$
1070  X=X+7 : GOSUB *DISP  ' -- [f·2]-[f·5]
1080 NEXT I
1090 READ A$
1100  X=X+10 :GOSUB *DISP  ' -- [f·6]
1110 FOR I=1 TO 4 :READ A$
1120  X=X+7 : GOSUB *DISP  ' -- [f·7]-[f·10]
1130 NEXT I
1140 '---- Key Select ----
1150 *KEYS
1160 LOCATE 0,2:PRINT SPACE$(6)
1170 ON KEY GOSUB *SAKUSEI,*TSUIKA,*HENKOU,*SAKUJO,*CHIKAN,
      *YOMIKOMI,*KENSAKU,*TOUROKU,*SHOUKYO,*SHURYOU
1180 FOR I=1 TO 10 : KEY(I) ON : NEXT I
1190 '---- Display Prompt ----
1200 LOCATE 0,0
1210 PRINT "処理項目を選んでください。
1220 LOCATE 0,2:K$=INPUT$(1)
1230 GOTO *KEYS
1240 '====== SUBROUTINES ======
1250 *SAKUSEI
1260 PRINT "作 成" : K$=INPUT$(1) :RETURN
1270 *TSUIKA
1280 PRINT "追 加" : K$=INPUT$(1) :RETURN
1290 *HENKOU
1300 PRINT "変 更" : K$=INPUT$(1) :RETURN
1310 *SAKUJO
1320 PRINT "削 除" : K$=INPUT$(1) :RETURN
1330 *CHIKAN
1340 PRINT "置 換" : K$=INPUT$(1) :RETURN
1350 *YOMIKOMI
1360 PRINT "読 込" : K$=INPUT$(1) :RETURN
1370 *KENSAKU
1380 PRINT "検 索" : K$=INPUT$(1) :RETURN
1390 *TOUROKU
1400 PRINT "登 録" : K$=INPUT$(1) :RETURN
1410 *SHOUKYO
1420 PRINT "消 去" : K$=INPUT$(1) :RETURN
1430 *SHURYOU
1440 PRINT "終 了" : KEY OFF :CLS:CONSOLE 0,25,1,0
1450 END
1460 *DISP   ' Function Key Area Dispaly
1470 LOCATE X,Y : PRINT A$;
1480 COLOR@(X,Y)-(X+5,Y),4
1490 RETURN
1500 DATA 作 成,追 加,変 更,削 除,置 換
1510 DATA 読 込,検 索,登 録,消 去,終 了
```

9-4　漢字フォントパターン読み出し

インタラプトコールを使って漢字コードに対応するフォントパターンを読み出してみましょう。日本語コードは，全角が15×16ドット，半角が7×16ドット，¼角が6×8ドットとなっています。また，これらのパターンを読び出す際に必要なバッファの大きさは表9-4のように決められています。

	フォントパターン・ドット数	フォントバッファ・バイト数
全　角	15 × 16	32 + 2
半　角	7 × 16	16 + 2
¼　角	6 × 8	8 + 2

表9-4 漢字のタイプとリードバッファ

では次に，読み出し方を述べます。

```
; KANJI FONT READ
;
MOV AH,14H          ; BIOS コード
MOV BX,BUFSEG       ← バッファの先頭アドレル（セグメント）
MOV CX,BUFOFF       ← バッファの先頭アドレス（オフセット）
MOV DX,JCODE        ← 文字コードを指定
```

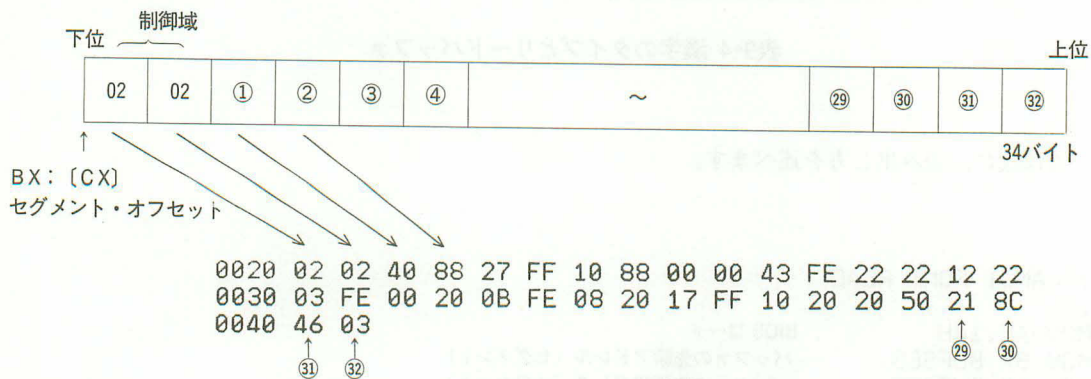
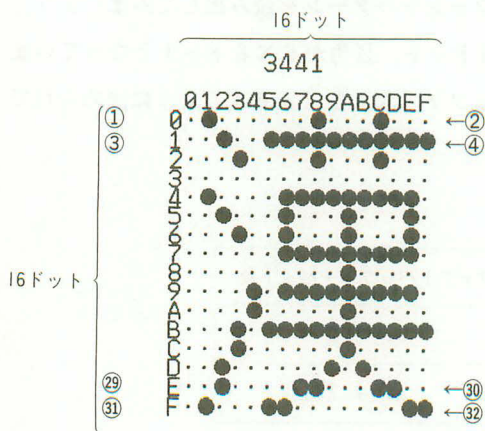
・全角………DX＝JISコード
・半角………DX＝半角コード
・¼角………DX＝¼角コードの下位8バイト

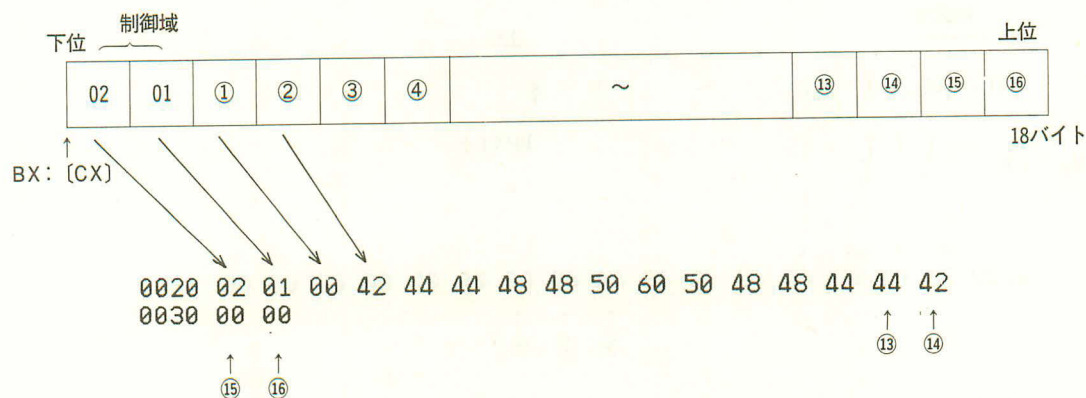
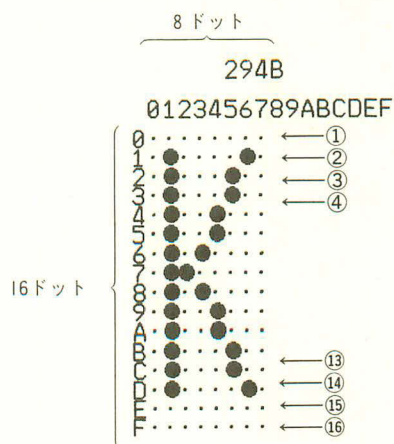
```
INT 18H             ; INT CALL
;
```

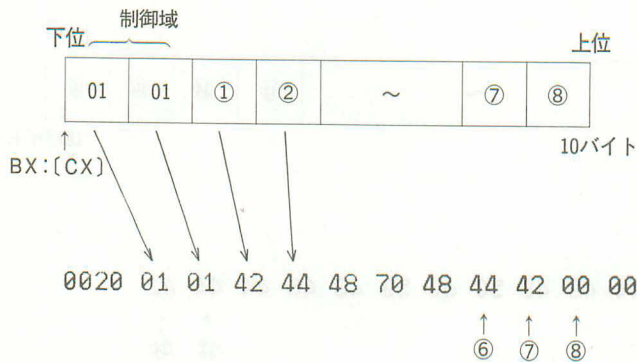
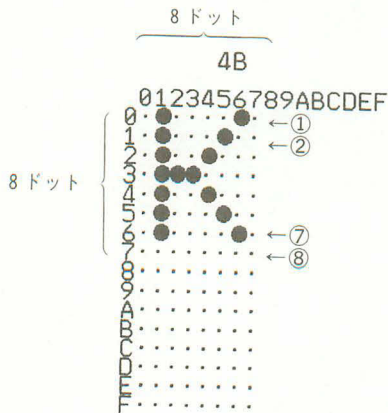
これを実行後，使用したレジスタ（AH，BX，CX，DX）およびSI以外のレジスタは保障され，次の形式でBX，CXで指定したフォントパターンバッファに書き込まれます。ただし，バッファの最初の2バイトは制御域となっており，実際のフォントは3バイト目から格納されます。

以下，セグメント1F00H，オフセット20Hとして全角，半角，¼角のフォントパターンを読み出す方法を具体例を示して紹介します。

① 全角

```
0020 02 02 40 88 27 FF 10 88 00 00 43 FE 22 22 12 22
0030 03 FE 00 20 0B FE 08 20 17 FF 10 20 20 50 21 8C
0040 46 03
```

```
0000 B414        MOV    AH,14
0002 BB001F      MOV    BX,1F00 ←──セグメント
0005 B92000      MOV    CX,0020 ←──オフセット
0008 BA4134      MOV    DX,3441 ←──「漢」全角
000B CD18        INT    18
000D C3          RET
```

② 半角

```
0000 B414          MOV    AH,14
0002 BB001F        MOV    BX,1F00
0005 B92000        MOV    CX,0020
0008 BA4B80        MOV    DX,804B ←「K」半角
000B CD18          INT    18
000D C3            RET
```

③ ¼角

```
0000 B414          MOV     AH,14
0002 BB001F        MOV     BX,1F00
0005 B92000        MOV     CX,0020
0008 BA4B00        MOV     DX,004B ←「K」¼角
000B CD18          INT     18
000D C3            RET
```

読み出したフォントパターンは一般には，グラフィック VRAM に移送してディスプレイに表示します。それでは，それを実際に行ってみましょう。次のプログラムは，BLUE のグラフィック画面の左上から横に「漢字表示 DEMO」と表示するものです。

漢字フォントG-VRAM書き込み

```
1 'save "FONT.RAM"
100 ' Read Font and Display on G-VRAM
110 ' ---  FT=0:CALL FT ---
120 DEF SEG=&H1F00
130  FOR I=0 TO &H7A : READ D$
140    D=VAL("&H"+D$) : POKE I,D
150  NEXT I
160 END
170 DATA 1E,0E,1F,B8,00,A8,8E,C0,B4,14,BB,00,1F,B9,49,00
180 DATA BE,6B,00,33,FF,BD,08,00,8B,14,56,CD,18,5E,E8,0A
190 DATA 00,46,46,83,C7,02,4D,75,EF,1F,CF,50,53,51,56,57
200 DATA B9,10,00,BE,49,00,8B,44,02,26,89,05,46,46,83,C7
210 DATA 50,E2,F3,5F,5E,59,5B,58,C3,00,00,00,00,00,00,00
220 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
230 DATA 00,00,00,00,00,00,00,00,00,00,00,41,34,7A,3B,3D
240 DATA 49,28,3C,44,23,45,23,4D,23,4F,23,00,00,00,00,00
```

マシン語ルーチンのソースリスト

```
                        ;************************
                        ;* KANJI FONT READ
                        ;* AND DISPLAY
                        ;************************
                        ;
0000 1E                 START:  PUSH DS               ; SAVE DS
0001 0E                         PUSH CS
0002 1F                         POP DS                ; DS=CS
0003 B800A8                     MOV AX,0A800H                 ; GVRAM BLUE SEGMENT
0006 8EC0                       MOV ES,AX
0008 B414                       MOV AH,14H                    ; BIOS CODE
000A BB001F                     MOV BX,1F00H                  ; FONT BUFFER SEGMENT
000D B94900                     MOV CX,OFFSET BUFF            ; FOND BUFFER OFFSET
0010 BE6B00                     MOV SI,OFFSET KCODE           ; KANJI CODE AEAR
0013 33FF                       XOR DI,DI                     ; GVRAM OFFSET ADDRESS
0015 BD0800                     MOV BP,08H                    ; LOOP COUNTER
                        ;
0018 8B14               LOOP1:  MOV DX,[SI]                   ; DX=KANJI CODE
001A 56                         PUSH SI
001B CD18                       INT 18H                       ; INT CALL
001D 5E                         POP SI
001E E80A00 002B                CALL DISP                     ; GVRAM WRITE
0021 46                         INC SI
0022 46                         INC SI
0023 83C702                     ADD DI,02H                    ; X=X+2
0026 4D                         DEC BP
0027 75EF   0018                JNZ LOOP1
0029 1F                         POP DS                        ; RESTORE DS
                        ;
002A CF                         IRET                          ; BACK TO BASIC
                        ;
002B 50                 DISP:   PUSH AX                       ; SAVE REGISTERS
002C 53                         PUSH BX
002D 51                         PUSH CX
002E 56                         PUSH SI
002F 57                         PUSH DI
                        ;
```

```
0030 B91000                      MOV CX,10H              ; CX=16 WORDS
0033 BE4900                      MOV SI,OFFSET BUFF
0036 8B4402             LOOP2:   MOV AX,02H[SI]
0039 268905                      MOV ES:[DI],AX          ; WRITE 2 BYTES
003C 46                          INC SI
003D 46                          INC SI
003E 83C750                      ADD DI,50H              ; SKIP 80 BYTES
0041 E2F3         0036           LOOP LOOP2              ; Y=Y+80
                        ;
0043 5F                          POP DI                  ; RESTORE REGISTERS
0044 5E                          POP SI
0045 59                          POP CX
0046 5B                          POP BX
0047 58                          POP AX
0048 C3                          RET                     ; RETUREN TO MAIN
                        ;
                        ;
                        ;---------------------------------------------
                        ; DATA SEGMENT
                        ;---------------------------------------------
                        ;
  0049                  DA       EQU OFFSET $
                                 DSEG
                                 ORG DA
                        ;
0049                    BUFF     RS 34                   ; RESERVE 34 BYTES
006B 41347A3B3D49       KCODE    DW 3441H,3B7AH,493DH,3C28H
     283C
0073 44234D234523                DW 2344H,234DH,2345H,234FH
     4F23
                        ;
```

9-5 漢字フォントパターンの拡大表示

漢字フォントパターンの読み出し方が分かったところで，BASICとマシン語によりフォントパターンを画面に拡大して表示してみましょう。RUNして，漢字コードを16進数で入力すると，そのフォントパターンが表示されます。

フォントパターン拡大表示

```
1 'save "kdisp.bas"
100 ' Kanji Font Display
110 CLEAR ,&H1F00 : DEF SEG=&H1F00 ' Program Segment
120 WIDTH 80,25 : DEFINT A-Z
130 POFF=0 : FOFF=&H20
140 P=15 : K=31 : DIM P$(P),K(K)
150 FOR I=0 TO P : READ P$(I) : NEXT I
160 DATA ····,···●,··●·,··●●,·●··,·●·●,·●●·,·●●●
170 DATA ●···,●··●,●·●·,●·●●,●●··,●●·●,●●●·,●●●●
180 ' --- Kanji Fond Read Sub ----
190 FOR I=&H0 TO &HF              ' Program Offset=0
200   READ D$:D=VAL("&H"+D$)
210   POKE I,D
220 NEXT I
230 DATA C4,37         ' LES SI,[BX]
240 DATA 26,8B,14      ' MOV DX,ES:[SI]  ··· DX=Kanji Code
250 DATA B4,14         ' MOV AH,14H      ··· BIOS cmd
260 DATA BB,00,1F      ' MOV BX,1F00H    ··· Font Segment
270 DATA B9,20,00      ' MOV CX,0020H    ··· Font Offset
280 DATA CD,18         ' INT 18H         ··· Call Read Font
290 DATA CF            ' IRET            ··· Back to BASIC
300 ' --- Kanji Code Input & Read  ---
310 PRINT "=== Kanji Font Display ==="
320 INPUT "Enter Kanji Code ";KC$ : KC=VAL("&H"+KC$)
330 FOR I=0 TO K+2 : POKE FOFF+I,0 : NEXT I   ' Buffer Clear
340  CALL POFF(KC)  ' Read Fond
350 ' --- Store Font Data in k(k) ---
360 FOR I=0 TO K : K(I)=PEEK(FOFF+I+2) :NEXT I
370 '
380 ' --- Display ----
390  TP=PEEK(FOFF+1)
400 LOCATE 6,2 : PRINT HEX$(KC)
410 LOCATE 1,4 : PRINT "0123456789ABCDEF"
420 FOR I=0 TO P
430  LOCATE 0,I+5 : PRINT HEX$(I);
440  FOR J=0 TO TP-1
450   K$=RIGHT$("0"+HEX$(K((I*2+J)/(3-TP))),2)
460    PL=VAL("&H"+LEFT$(K$,1))
470    PR=VAL("&H"+RIGHT$(K$,1))
480    LOCATE J*8+1,I+5
490    PRINT P$(PL);P$(PR);
500  NEXT J
510 NEXT I
520 END
```

9-6 漢字・JISコード対応表示

PC-9801には日本語BASICのシステムが別売されていますが，これがない場合，いちいち漢字コードを調べるのは大変です。そこで，G-VRAMに漢字とそのJISコードを対応させて表示する方法をちょっとしたアイディアとして紹介しましょう。

次のプログラムは，漢字の最初の「ア」の部分をそのJISコードとともに表示するものです。GVRAMに表示しますので，テキスト画面で漢字入力する際に便利だと思います。これを「ア～ン」までに拡張して，例えば「ア」を入力すると，その漢字とコードが表示されるようにプログラミングすると一段と使い易くなることでしょう。

漢字・JISコード対応表

```
1 'save "Kdisp"
1000 SCREEN 3,0 :ROLL 399 :ROLL 1
1010 X=20:Y=0:N=0 ' Locate x,y
1020 FOR J=&H3021 TO &H3049 ' ア ...
1030   FOR I=1 TO 4  ' JIS Code print
1040   K$=HEX$(J):K$=MID$(K$,I,1)
1050    K=VAL(K$):K=K+&H130
1060    IF K$="A" THEN K=&H141
1070    IF K$="B" THEN K=&H142
1080    IF K$="C" THEN K=&H143
1090    IF K$="D" THEN K=&H144
1100    IF K$="E" THEN K=&H145
1110    IF K$="F" THEN K=&H146
1120   PUT (X+16*I-20,Y),KANJI(K),PSET,5,0:NEXT I
1130      PUT(X,Y+7),KANJI(J),PSET,6,0 ' Kanji print
1140  Y=Y+24:N=N+1:IF N>14 THEN Y=0:X=X+80:N=0
1150 NEXT J
```

〈出力例〉

```
3 0 2 1    3 0 3 0    3 0 3 F
亜         旭         或
3 0 2 2    3 0 3 1    3 0 4 0
唖         葦         粟
3 0 2 3    3 0 3 2    3 0 4 1
娃         芦         袷
3 0 2 4    3 0 3 3    3 0 4 2
阿         鯵         安
3 0 2 5    3 0 3 4    3 0 4 3
哀         梓         庵
3 0 2 6    3 0 3 5    3 0 4 4
愛         圧         按
3 0 2 7    3 0 3 6    3 0 4 5
挨         斡         暗
3 0 2 8    3 0 3 7    3 0 4 6
姶         扱         案
3 0 2 9    3 0 3 8    3 0 4 7
逢         宛         闇
3 0 2 A    3 0 3 9    3 0 4 8
葵         姐         鞍
3 0 2 B    3 0 3 A    3 0 4 9
茜         虻         杏
3 0 2 C    3 0 3 B
穐         飴
3 0 2 D    3 0 3 C
悪         絢
3 0 2 E    3 0 3 D
握         綾
3 0 2 F    3 0 3 E
渥         鮎
```

9-7 漢字フォントをビットイメージで出力

せっかく漢字フォントパターンの読み出し方が分かったのですから，グラフィック画面表示のみならず，プリンタにも出力してみましょう。通常，漢字をプリントアウトするにはプリンタに漢字ROMが付いていなければなりません。しかし，ビットイメージ（ビットドット対応グラフィック）が出力できるプリンタであれば，ここに示すルーチンで簡単に漢字出力ができます。ここでは，PC-8821を例にとり話を進めていきますが，その他のプリンタ（PC-8023C）などでも考え方は同じです。

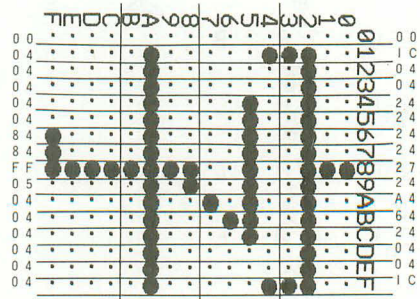
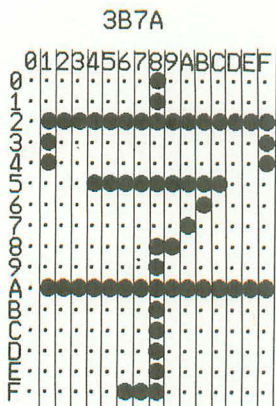
漢字の「字」を出力することを考えてみましょう。その手順は次のようになります。

① インタラプトコールで漢字フォントパターンを読み出し，バッファに格納する。

② そのパターンをビットイメージデータに変換する。

③ ビットイメージデータをプリンタに出力する。

①と③は簡単ですが，②がちょっとやっかいです。PC-8821 および PC-8822 では，16ビットドットのデータを出力するには次のようにします。

また漢字フォントは，縦方向にスライスして2バイトの値に変換しなくてはなりません。その変換法を「字」というフォントパターンを分析して解説しましょう。

フォントパターンを縦にスライスする　　　ビットイメージデータに変換する

ビットイメージデータは，00，00，1C，04～となります。BASICで「字」をプリントアウトするには次のようになります。

BASICによる「字」出力

```
1 ' save "kanji.bas"
10 LPRINT CHR$(27);"I0016";
20  FOR I=1 TO 32
30    READ K$ : K=VAL("&H"+K$)
40    LPRINT CHR$(K);
50  NEXT I
60 DATA 00,00,1C,04,04,04,04,04,24,04,24,04,24,84,24,84
70 DATA 27,FF,24,05,A4,04,64,04,24,04,04,04,04,04,1C,04
```

このフォントパターンからビットイメージデータに図を書いていちいち変換していたのでは頭が変になります。そこで，それをPC-9801にやってもらうプログラムが次のものです。これは前出の①，②，③の処理をすべて行います。使い方は，BASICから漢字コード（ただし，全角のもの）をパラメータとしてコールすればOKです。

フォントのビットイメージ出力

```
1 'save "KFP"
100 ' Kanji Font Print
110 ' ---  KF=0:CALL KF ---
120 DEF SEG=&H1F00
130  FOR I=0 TO &HD0 : READ D$
140    D=VAL("&H"+D$) : POKE I,D
150  NEXT I
160 END
170 DATA C4,37,26,8B,14,1E,0E,1F,B4,14,8C,DB,B9,AA,00,CD
180 DATA 18,BF,D2,00,BE,AA,00,46,46,33,DB,B1,01,56,57,53
190 DATA 51,33,C0,51,B9,08,00,89,05,47,47,E2,FA,59,BF,D2
200 DATA 00,32,ED,8A,34,8A,54,01,D3,E2,9F,F6,C4,01,75,04
210 DATA B0,00,7A,02,B0,01,88,05,47,83,C6,02,FE,C5,80,FD
220 DATA 10,75,E0,BF,D2,00,B9,02,00,E8,24,00,88,26,D1,00
230 DATA E8,1D,00,8A,C4,B2,10,F6,E2,02,06,D1,00,E8,31,00
240 DATA E2,E7,59,5B,5F,5E,FE,C1,43,83,FB,10,75,9F,1F,CF
250 DATA 51,32,E4,B1,01,8A,05,0A,C0,74,03,E8,0C,00,02,E0
260 DATA 47,FE,C1,80,F9,05,75,ED,59,C3,8A,C1,BB,CC,00,D7
270 DATA C3,1E,16,1F,B4,11,CD,1A,1F,C3,00,00,00,00,00,00
280 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
290 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,01,02,04
300 DATA 08,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
```

使い方

```
1 'save "KFP.BAS"
10 CLEAR ,&H1F00:DEF SEG=&H1F00:KF=0
20  FOR I=1 TO 8
30    READ KC$:KC%=VAL("&H"+KC$)
40    LPRINT CHR$(27);"I0016";
50      CALL KF(KC%)
60  NEXT I
```

```
70 END
80 DATA 3441,3B7A,493D,3C28,2344,2345,234D,234F
```

マシン語部分のソースリスト

```
                        ;**********************
                        ;* KANJI FONT PRINT
                        ;* IN BIT IMAGE
                        ;**********************

0000 C437               START:  LES SI,[BX]     ; GET PARA FROM BASIC
0002 268B14                     MOV DX,ES:[SI]  ; DX=KANJI CODE(16 DOT)
0005 1E                         PUSH DS         ; SAVE DS FOR INT CALL
0006 0E                         PUSH CS
0007 1F                         POP DS
                        ;
0008 B414               FONT:   MOV AH,14H      ; BIOS COMMAND
000A 8CDB                       MOV BX,DS       ; SEGMENT
000C B9AA00                     MOV CX,OFFSET BUFF      ; FONT BUUFER
000F CD18                       INT 18H         ; INT CALL
                        ;
0011 BFD200             GET:    MOV DI,OFFSET PWORK     ; BIT STORE WORK
0014 BEAA00                     MOV SI,OFFSET BUFF
0017 46                         INC SI
0018 46                         INC SI          ;SI=FIRST BUFFER
0019 33DB                       XOR BX,BX       ; BX=OUTER LOOP C=0
001B B101                       MOV CL,1        ; SHIFT COUNTER
001D 56                 OUTLP:  PUSH SI         ; SAVE KANJI FONT ADD
001E 57                         PUSH DI         ; SAVE PWORK ADD
001F 53                         PUSH BX         ; SAVE OUTER C
0020 51                         PUSH CX         ; SAVE CL COUNTER
0021 33C0                       XOR AX,AX       ; CLEAR PWORK
0023 51                         PUSH CX
0024 B90800                     MOV CX,8
0027 8905               CLR:    MOV [DI],AX
0029 47                         INC DI
002A 47                         INC DI
002B E2FA          0027         LOOP CLR
002D 59                         POP CX
002E BFD200                     MOV DI,OFFSET PWORK
                        ;
0031 32ED                       XOR CH,CH       ; INNER LOOP COUNTER CH=0
0033 8A34               INLP:   MOV DH,[SI]     ; DX=FIRST DOT STRING
0035 8A5401                     MOV DL,01H[SI]
0038 D3E2                       SHL DX,CL       ; SHIFT LEFT CL CF=BIT
003A 9F                         LAHF            ; AH=CF
003B F6C401                     TEST AH,1       ; CF=1?
003E 7504          0044         JNE NEXT        ; IF CF=0 THEN AL=0
0040 B000                       MOV AL,0        ; IF CF=1 THEN AL=1
0042 7A02          0046         JP STR
0044 B001               NEXT:   MOV AL,1
0046 8805               STR:    MOV [DI],AL     ; STORE
0048 47                         INC DI
0049 83C602             NEXLP:  ADD SI,02H      ; SKIP 2 BYTES
004C FEC5                       INC CH          ; COUNT UP
004E 80FD10                     CMP CH,16       ; 16 LOOPS?
0051 75E0          0033         JNE INLP        ; INNER LOOP
```

```
                                ;
0053 BFD200             PICKAL: MOV DI,OFFSET PWORK
0056 B90200                     MOV CX,2            ; LOOP 2
0059 E82400        0080 PCHR:   CALL PICKUP
005C 8826D100                   MOV XLOW,AH         ; STORE LOW BYTE
0060 E81D00        0080         CALL PICKUP
0063 8AC4                       MOV AL,AH
0065 B210                       MOV DL,16           ; YLOW=Y*16
0067 F6E2                       MUL DL
0069 0206D100                   ADD AL,XLOW         ; AL=BYTE TO PRINTER
006D E83100        00A1         CALL PUTC           ; OUTPUT TO PRINTER
0070 E2E7          0059         LOOP PCHR
                                ;
0072 59                         POP CX              ; RESTORE CL
0073 5B                         POP BX
0074 5F                         POP DI              ; RESTORE KWORD ADD
0075 5E                         POP SI              ; RESTORE KANJI ADD
0076 FEC1                       INC CL              ; SHIFT BIT COUNT UP
0078 43                         INC BX              ; OUTER LOOP COUNT UP
0079 83FB10                     CMP BX,16
007C 759F          001D         JNE OUTLP
                                ;++++++++++++++++++++++++++++++++++++++++
007E 1F                         POP DS
007F CF                         IRET                ; BACK TO BASIC
                                ;++++++++++++++++++++++++++++++++++++++++
                                ;
0080 51                 PICKUP: PUSH CX
0081 32E4                       XOR AH,AH
0083 B101                       MOV CL,1            ; LOOP 4 COUNTER
0085 8A05               AD:     MOV AL,[DI]
0087 0AC0                       OR AL,AL
0089 7403          008E         JZ ZERO
008B E80C00        009A         CALL CALC           ; 1,2,4,8
008E 02E0               ZERO:   ADD AH,AL
0090 47                         INC DI
0091 FEC1                       INC CL
0093 80F905                     CMP CL,5
0096 75ED          0085         JNE AD
0098 59                         POP CX
0099 C3                         RET
                                ;
                                ;-----------------------------------------------
                                ;---SUBROUTINE----------------------------------
                                ;
009A 8AC1               CALC:   MOV AL,CL
009C BBCC00                     MOV BX,OFFSET TBL        ; 1,2,4,8
009F D7                         XLAT AL             ; AL=1,2,4,8
00A0 C3                         RET
                                ;
00A1 1E                 PUTC:   PUSH DS
00A2 16                         PUSH SS
00A3 1F                         POP DS
00A4 B411                       MOV AH,11H          ; BIOS CMD
00A6 CD1A                       INT 1AH             ; OUT 1 BYTE (AL)
00A8 1F                         POP DS
00A9 C3                         RET
                                ;
                                ;===============================================
                                ; DATA SEGMENT
                                ;===============================================
```

```
                        ;
 00AA                   DATA EQU OFFSET $
                        ;
                                DSEG
                                ORG DATA
                        ;
00AA                    BUFF    RS 34
00CC 0001020408         TBL     DB 00H,01H,02H,04H,08H
00D1                    XLOW    RS 1
00D2                    PWORK   RS 16
                        ;
                                END
```

9-8 任意のフォントの作成・出力

漢字の章の最後として，これまでのことがらをすべて1つのプログラムにまとめたものを紹介します。機能としては，画面上で16×16ドットの範囲で任意にフォントを作成・修正してそれをビットイメージでプリンタに出力できます。これにより，ユーザーは思いのままに文字やパターンが出力できます。また，作成中にパターンがG-VRAM上に表示され，パターンの確認ができたり，作成済のパターンをディスクにセーブすることもできます。

なお，このプログラムは，前出の漢字フォントパターンの拡大表示，G-VRAMへの書き込み，ビットイメージ出力を応用したものです。

使い方は次のとおりです。

1………Edit（編集）

① Edit New（新規作成）

1度にパターンを続けて作成できます。

② Edit Old（作成済データ編集）

ファイル名を入力。ただし，データの追加は不可。

2………Display Data（作成済データ表示）

ファイル名を入力すると，そのファイルの登録文字数とパターンが表示。

3………Print Out（プリンタ出力）

ファイル名を入力すると，そのファイルのすべてのパターンが続けてプリンタに出力されます。

次に出力サンプルを示しておきます。

PC-TechKnow9800

任意のフォント作成・出力

```
1 'save "Fedit
1000 CLEAR ,&H1F00 : DEF SEG=&H1F00 :SCREEN 0,0
1010 WIDTH 80,25 :DEFINT A-Z:ROLL 199
1020 K=31 : DIM K(K),P$(K):CONSOLE 0,25,0,0
1030 FOR I=0 TO 15:READ P$(I):NEXT I
1040 DATA ····,···●,··●·,··●●,·●··,·●·●,·●●·,·●●●
1050 DATA ●···,●··●,●·●·,●·●●,●●··,●●·●,●●●·,●●●●
1060 FOR I=0 TO &H9E:READ D$  ' Bit Image Print
1070 D=VAL("&H"+D$):POKE I,D  ' Machine Code
1080 NEXT I
1090 RESTORE 2420:FOR I=&H200 TO &H229:READ D$ 'G-VRAM
1100 D=VAL("&H"+D$):POKE I,D  ' Machine Code
1110 NEXT I
1120 PRINT "=== User-Defined Character ==="
1130 PRINT "1 ·· Edit  2 ·· Display Data  3 ·· Print Out"
1140 INPUT "Select No.(Press RETURN to END)";NO
1150  CLS:H$="0123456789ABCDEF"
1160 WK=0:BIT=&H0 :DT=&H250
1170 ON NO GOSUB *FEDIT,*DDISP,*POUT
1180  CLS :IF NO=0 THEN CONSOLE 0,25,1,0:END
1190 ERASE K:DIM K(K):GOTO 1120
1200 *FEDIT
1210 INPUT "1 ·· Edit New   2 ·· Edit Old";NW
1220 IF NW=1 THEN CN=0:GOTO 1300
1230 'FILES :PRINT
1240 INPUT "Enter File Name to Edit ";F$:BLOAD F$
1250 INPUT "Enter Character No. to Edit ";CN
1260 CLS
1270 J=WK:CAD=DT+CN*32 : FOR I=CAD TO CAD+31
1280  DP=PEEK(I):K(J)=DP:J=J+1
1290 NEXT I
1300 PRINT "==== Font Editor ===="
1310 PRINT" Press Cursor Key to Move"
1320 PRINT" Press SPACE BAR to Set/Reset Dot"
1330 PRINT" Press RETURN when Done"
1340  IF NW<>1 THEN GOSUB *DISP :GOTO 1390
1350 LOCATE 2,5:PRINT H$
1360 FOR I=0 TO 15
1370  LOCATE 1,I+6:PRINT HEX$(I);STRING$(16,"·")
1380 NEXT
1390 ' ---- Font Editor ----
1400 N=0:CR$=CHR$(13):SG=VARPTR(K(0),1):GV=&H200
1410 LT$=CHR$(28):RT$=CHR$(29)
1420 UP$=CHR$(30):DN$=CHR$(31)
1430 *START
1440   CALL GV(SG)  ' G-VRAM Display
1450 X=N MOD 16 : Y=N ¥ 16 :LOCATE X+2,Y+6
1460 K$=INPUT$(1)
1470  IF K$<>" " THEN *KINP
1480    IF (K(Y*2+X ¥ 8) AND 2^(7-X MOD 8))=0 THEN *DSET
1490    PRINT "·";
```

```
1500    K(Y*2+X ¥ 8)=K(Y*2+X ¥ 8) AND NOT(2^(7-X MOD 8)):
         GOTO *START
1510 *DSET
1520    PRINT "●";
1530    K(Y*2+X ¥ 8)=K(Y*2+X ¥ 8) OR (2^(7-X MOD 8)):
          GOTO *START
1540 *KINP
1550  IF K$=LT$ THEN N=N+1 :IF N>255 THEN N=255  :GOTO *START
1560  IF K$=RT$ THEN N=N-1 :IF N<0   THEN N=0    :GOTO *START
1570  IF K$=UP$ THEN N=N-16:IF N<0   THEN N=N+16 :GOTO *START
1580  IF K$=DN$ THEN N=N+16:IF N>255 THEN N=N-16 :GOTO *START
1590  IF K$=CR$ THEN *WFD
1600 GOTO *START
1610 '
1620 ' ---- Write Font Data----
1630  *WFD
1640  J=WK:CAD=DT+CN*32:FOR I=CAD TO CAD+31
1650  POKE I,K(J):J=J+1
1660 NEXT I
1670 CLS:ROLL 199
1680 INPUT " Continue (y/n)";CNT$
1690 IF CNT$="y" OR CNT$="Y" THEN IF NW=1 THEN CN=CN+1:
         ERASE K:DIM K(K):GOTO 1350 ELSE 1250
1700 IF CNT$<>"y" OR CNT$<>"Y" THEN IF NW=2 THEN 1740
1710 BYT=32*(CN+1)
1720 INPUT "Enter File Name to Save  ";F$:BSAVE F$,DT,BYT
1730   GOTO 1760
1740 GOSUB *ADDCAL:BYT=32*CNT
1750 PRINT "Now saving edited data..";F$:BSAVE F$,DT,BYT
1760 RETURN
1770 ' Data Display
1780 *DISP
1790 LOCATE 2,5 :PRINT H$
1800 FOR I=0 TO 15
1810  LOCATE 1,I+6:PRINT HEX$(I);
1820  FOR J=0 TO 1
1830  K$=RIGHT$("0"+HEX$(K((I*2+J)/1)),2)
1840   PL=VAL("&H"+LEFT$(K$,1))
1850   PR=VAL("&H"+RIGHT$(K$,1))
1860   LOCATE J*8+2,I+6
1870   PRINT P$(PL);P$(PR);
1880  NEXT J
1890 NEXT I
1900 RETURN
1910 ' Data Load and Dispaly
1920 *DDISP
1930 INPUT "Enter File Name to Load ";F$:BLOAD F$
1940  GOSUB *ADDCAL
1950 LOCATE 2,2:PRINT "CHR NOS.";0;"-";CNT-1
1960  FOR CN=0 TO CNT-1
1970 LOCATE 2,3:PRINT "Now Showing NO.";CN
1980    J=WK:CAD=DT+CN*32 : FOR I=CAD TO CAD+31
1990    DP=PEEK(I):K(J)=DP:J=J+1
2000    NEXT I
2010  GOSUB *DISP
2020  SG=VARPTR(K(0),1):GV=&H200:CALL GV(SG)
2030  PRINT:PRINT:INPUT "Press RETURN ",A$
2040   ROLL 199:NEXT CN
2050 RETURN
2060 ' --- PRINT OUT ----
```

```
2070 *POUT
2080 INPUT "Enter File Name to Print ";F$:BLOAD F$
2090  GOSUB *ADDCAL
2100  BIT=&H0
2110  FOR CN=0 TO CNT-1
2120    J=&H9F:CAD=DT+CN*32 : FOR I=CAD TO CAD+31
2130    DP=PEEK(I):POKE J,DP:J=J+1
2140    NEXT I
2150   LPRINT CHR$(27);"I0016"; ' Bit image
2160   CALL BIT
2170  NEXT CN :LPRINT
2180 RETURN
2190 '--- Address Calc ----
2200 *ADDCAL
2210 OPEN F$ FOR INPUT AS #1
2220  C=0:A$=INPUT$(8,#1)
2230  AD=VARPTR(#1)+32:SG=VARPTR(#1,1):DEF SEG=SG
2240  AD=PEEK(AD)+PEEK(AD+1)*256
2250  SA=PEEK(AD)+PEEK(AD+1)*256
2260  EA=PEEK(AD+2)+PEEK(AD+3)*256
2270  CNT=(EA-SA)/32
2280  CLOSE #1:DEF SEG=&H1F00
2290 RETURN
2300 ' ---- Bit Image Print ----
2310 DATA 1E,0E,1F,BF,C0,00,BE,9F,00,33,DB,B1,01,56,57,53
2320 DATA 51,33,C0,51,B9,08,00,89,05,47,47,E2,FA,59,BF,C0
2330 DATA 00,32,ED,8A,34,8A,54,01,D3,E2,9F,F6,C4,01,75,04
2340 DATA B0,00,7A,02,B0,01,88,05,47,83,C6,02,FE,C5,80,FD
2350 DATA 10,75,E0,BF,C0,00,B9,02,00,E8,24,00,88,26,BF,00
2360 DATA E8,1D,00,8A,C4,B2,10,F6,E2,02,06,BF,00,E8,31,00
2370 DATA E2,E7,59,5B,5F,5E,FE,C1,43,83,FB,10,75,9F,1F,CF
2380 DATA 51,32,E4,B1,01,8A,05,0A,C0,74,03,E8,0C,00,02,E0
2390 DATA 47,FE,C1,80,F9,05,75,ED,59,C3,8A,C1,BB,9A,00,D7
2400 DATA C3,1E,16,1F,B4,11,CD,1A,1F,C3,00,01,02,04,08,00
2410 ' --- G-VRAM Display ---
2420 DATA 56,57,1E,C5,37,8B,04,8E,D8,B8,00,B8,8E,C0,BF,00
2430 DATA 16,B9,10,00,31,F6,8A,04,8A,64,02,26,89,05,83,C6
2440 DATA 04,83,C7,50,E2,F0,1F,5F,5E,CF,00,00,00,00,00,00
```

注）作成中のフォントのG-VRAMへの表示を640×400モードにするには次の行を変更して下さい。

1000……SCREEN 0,0 ⟶ SCREEN 3, 0

1010……ROLL 199 ⟶ ROLL 399

1670……ROLL 199 ⟶ ROLL 399

2040……ROLL 199 ⟶ ROLL 399

2430……DATA 16 ⟶ DATA 2F

第 10 章　USR関数・CALL文とマシン語

10-1　マシン語ルーチンの呼び方

10-2　マシン語ルーチンの実行と引数の受け渡し方

- 10-2-1　引数がない場合
- 10-2-2　USR関数の引数
- 10-2-3　CALL文の引数

10-3　結果の戻し方

- 10-3-1　USR関数の場合
- 10-3-2　CALL文の場合

10-4　BASIC＋マシン語ルーチン

- 10-4-1　サウンドビープ
- 10-4-2　小文字・大文字変換
- 10-4-3　最大値を求める
- 10-4-4　文字列を逆に表示
- 10-4-5　ROLL200&ROLL400
- 10-4-6　アドレスサーチ

第10章　USR関数・CALL文とマシン語

N_{88}-BASIC (86) は命令の豊富さ，実行の速さなどを考えるとBASICインタプリタとしては最強のものの1つと言えます。しかし，BASICよりもさらに高速処理を必要とするもの(例えばサーチやソート)，さらにBASICではサポートされていない機能を引き出すためにはマシン語が必要となってきます。本書ではいろいろマシン語ルーチンが紹介されていますが，この章ではN_{88}-BASIC (86)とマシン語との接点となるUSR関数とCALL文について説明します。BASICからマシン語に引数(パラメータ)を受け渡す方法やそれをマシン語ルーチンで処理してBASICに戻すことなどを述べます。

10-1　マシン語ルーチンの呼び方

BASICからマシン語ルーチンをコールしたい場合，N_{88}-BASIC (86) ではUSR関数またはCALL文を用います。USR関数は，USRの後に0から9までの数値をつけることによって一度に最高10種類まで定義することができます。CALL文はコールするアドレスを指定することにより，いくつも定義できます。また，マシン語ルーチンに渡す引数（パラメータ）は，USR関数では1つ，CALL文では0から制限なしとなっています。

次に両者の設定法を述べます。

USR関数

DEF　SEG＝セグメントアドレス（実行セグメント）

DEF　USRn＝オフセットアドレス（実行開始番地）

変数名＝USRn（数値 or 変数 or 式）

↓ 型（0～9）　　↓ 引数

CALL文

DEF　SEG＝セグメントアドレス（実行セグメント）

変数名＝オフセットアドレス（実行開始番地）

CALL　変数名（変数名，変数名，………）

↓ 引数（整数型）

このUSR関数とCALL文との実行のしかたと機能の違いをまとめたのが表10-1です。

設定法 ＼ 種類	USR関数	CALL文
セグメント指定 (1F00Hのとき)	DEF SEG=&H1F00	DEF SEG=&H1F00
オフセット指定(20Hのとき)	DEF USR=&H20	SUB=&H20
引数設定と実行	• 引数は1個だけ • 引数はかならず必要 (不要のときはダミーを入れる) • 直接・関接に設定可 A=USR(100) B=1：A=USR(B) • 引数は整数, 単精度, 倍精度, 文字型のの4種 • 引数は加減算など演算が可能	• 引数はいくつでも可 • 引数がなくても可 • 引数は変数に代入する必要がある A%=5：CALL SUB(A%) • 引数は整数, 単精度, 倍精度, 文字型の4種 (混合可) • 引数は演算不可
注意点	マシン語ルーチンでALの値を変更してBASICに戻すと Type mismatch エラーとなる。しかし, XOR AX,AX としてIRET するとエラーとはなりません。	ALの値を変更しても可
BASICに戻るとき	IRET	IRET

表10-1 USR関数・CALL文の比較

次に実際の使用例をあげます。

型	実行開始番地	引数	実行
0	DEF USR=&H0	ダミー	A=USR(0)
1	DEF USR1=&H20	整数型	A%=USR1(&H100)
2	DEF USR2=&H50	実数型演算	B=1：C=2 A=USR2(B+C)
3	DEF USR3=&H100	文字列間接	B$= "ABC" A$=USR3(B$)
4	DEF USR4=&H120	文字列直接	A$=USR4("XYZ")
5	DEF USR5=&H150	文字列演算	B$="ABC" A$=USR5(B$+"XYZ")

USR関数 (DEF SEG=&H1F00) の使用例

実用開始番地	引　　数	実　　行
SUB=0	な　　し	CALL SUB
SUB=&H20	整数型1 個	A%=1 CALL SUB(A%)
SUB=&H50	単精度型2 個	A=100：B=200 CALL SUB(A, B)
SUB=&H120	倍精度型3 個	A#=60000 B#=70000 C#=80000 CALL SUB(A#, B#, C#)
SUB=&H140	文字型1 個	A$="ABC" CALL SUB(A$)
SUB=&H160	文字型2 個	A$="ABC" B$="XYZ" CALL SUB(A$, B$)
SUB=&H200	文字型2 個+ 整数型1 個	A$="ABC" B$="XYZ" C%=&H1234 CALL SUB(A$, B$, C%)

CALL文(DEF SEG=&H1F00)の使用例

USR関数とCALL文の決定的な違い。

次に2つのプログラムを見較べて下さい。

USR関数

```
10 DEF SEG=&H1D00
20 DEF USR=&H0
30 DEF SEG=&H1F00
40 A=USR(0)
```

CALL文

```
10 DEF SEG=&H1D00
20 SUB=&H0
30 DEF SEG=&H1F00
40 CALL SUB
```

一見，どちらも同じ機能を果たしているようですが，実は大きな違いがあります。USR 関数では物理アドレス 1 D 000 H 番地から実行しますが, CALL 文では 1 F 000 H 番地から実行します。つまり，USR 関数では，セグメントの指定が変わっても，DEF　USR が指定されたときのセグメントが選択されます。これは，DEF USR が指定されると，ワークエリアにそのときのオフセットとセグメントが書き込まれるためです。ですから，セグメントをいくつも切り換えるようなプログラムでは, 実行するアドレスを正確に指定するためには, いつも対で指定する (DEF SEG＝&H 1 D 00：DEF　USR＝&H 0) と良いでしょう。

このことは考え方によっては，USR 関数でファーコール（セグメント間コール）が行えるということです。

これに対して CALL 文では，セグメントの指定が変われば，それからのオフセット番地から実行することになります。まあ，これが普通の考え方ですが。

それではサンプルを示しながら具体的に説明をしていきましょう。

10-2　マシン語ルーチンの実行と引数の受け渡し方

10-2-1 引数がない場合

まずは，BASIC からただ単にマシン語ルーチンを呼ぶためのものです。引数はありません。次の例はマシン語による簡単なタイマー（時間待ち）ルーチンで，実行すると 2 秒間ウエイトした後，BASIC に戻ってきます。

USR関数による実行

```
1 ′ save "SAMP1"
100 DEF SEG=&H1F00
110 DEF USR=&H0
120 TIME$="00:00:00"
130 A=USR(0)
140 PRINT TIME$
150 END
```

CALL文による実行

```
1 ′ save "SAMP2"
100 DEF SEG=&H1F00
110 SUB=&H0
120 TIME$="00:00:00"
130 CALL SUB
140 PRINT TIME$
150 END
```

タイマールーチン（2 秒間ウエイト）

```
                                ;=================
                                ; TIMER 2 SECONDS
                                ;=================
                                        ORG 0
                                ;
0000 B8FF06                     TIMER:  MOV AX,06FFH
0003 BBD000                     OTLOOP: MOV BX,00D0H
0006 4B                         INLOOP: DEC BX
0007 83FB00                             CMP BX,0
000A 75FA           0006                JNE INLOOP
000C 48                                 DEC AX
000D 3D0000                             CMP AX,0
0010 75F1           0003                JNE OTLOOP
0012 CF                         BACK:   IRET
                                ;
```

10-2-2 USR関数の引数

マシン語ルーチンがUSR関数から呼ばれたときは，引数の型がALレジスタに入ります。その値と型は表10-2-2のとおりです。

ALレジスタ	型	例
AL = 2	整　数　型	X％=128
AL = 4	単精度実数型	Y！=789.12
AL = 8	倍精度実数型	Z＃=12345678
AL = 3	文　字　型	A＄="XYZ"

表10-2-2 USR関数の型

このALの値によって，マシン語ルーチンの中で引数の型をチェックすることができます。例えば，文字列を受け渡したいときに，数値が渡されては困る場合に，ALの値を調べて文字型(3)でなければ何も処理をせずBASICに戻すときなどに使います。

```
例)       CMP AL,3H
          JNE BACK
             .
             .
             .
   BACK:  IRET
```

さて引数(またはその情報)は，BASICが使う8バイトのワークエリアに入ります。これは浮動小数点アキュムレーター (Floating Point Accumulator=FAC) と呼ばれています。

引数が数値の場合にはDS：BXにFACの5バイト目またはFACの先頭アドレスが入っています。実際の引数は，このFACの中に各型に応じて次のように格納されて渡されます。

● 整数型のとき

FACの5バイト目を〔BX〕レジスタが指しており，そこに引数の下位8ビットが入り，6バイト目に上位8ビットが入ります。

● 単精度実数のとき

FACの5バイト目を〔BX〕レジスタが指しており，5バイト目から8バイト目までの4バイトに内容が入ります。FACの8バイト目は指数部になっており，(指数−128)の値が入ります。小数点は仮数部の最上位ビットの左にあると想定します。7バイト目は仮数部の最高部7ビットを保持します。このバイトの最上位ビットは符号を示し，0で正，1で負を表します。6バイト目と5バイト目は，それぞれ仮数部の中部と最低部の8ビットを保持します。

● 倍精度実数型のとき

FACの1バイト目を〔BX〕レジスタが指しています。5バイト目から8バイト目までは単精度実数型と同じように指数部と仮数部の上位3バイトが入ります。1バイト目から4バイト目には仮数部の下位4バイトが入ります。

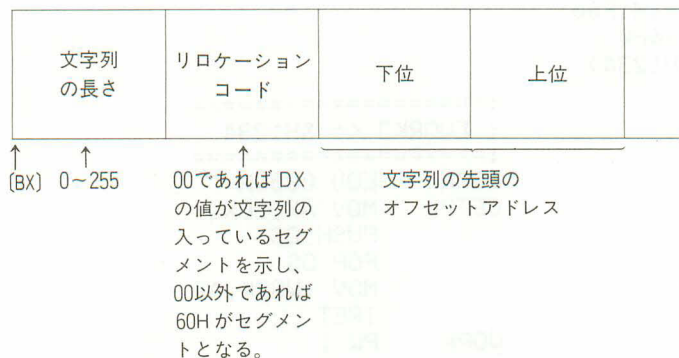
● 文字列のとき

引数が文字列の場合には〔BX〕レジスタがFACの5バイト目を指し，そこにストリング・ディスクリプタ（文字列の格納情報）の先頭アドレスがセグメントベースとオフセットの形態で入れられています。

FACはセグメント60H，オフセット1416H～141DHにあり，引数の格納状態は図10-2-2のようになります。

値 レジスタの FAC	引数の型	整　数　型	単精度実数型	倍精度実数型	文　字　型
	DS	FACのセグメント(60H)			
	BX	FACの5バイト目(141AH)		FACの1バイト目(1416H)	FACの5バイト目(141AH)
	AL	2	4	8	3
1st	1416H			仮数（最下位8ビット）	
2nd	1417H			仮数（第6位8ビット）	
3rd	1418H			仮数（第5位8ビット）	
4th	1419H			仮数（第4位8ビット）	
5th	141AH	下位バイト	仮数（下位8ビット）	仮数（第3位8ビット）	ストリングディスクリプタのオフセット下位バイト
6th	141BH	上位バイト	仮数（中位8ビット）	仮数（第2位8ビット）	ストリングディスクリプタのオフセット上位バイト
7th	141CH		仮数（上位8ビット）	仮数（最上位8ビット）	セグメント下位バイト
8th	141DH		指　数　部	指　数　部	セグメント上位バイト

図10-2-2 USR関数の引数格納状態

ストリングディスクリプタは次のようになっています。

以上の情報をもとにマシン語ルーチンで引数を受け取れば良いことになります。以下，数値（整数型）と文字の場合について例を示します。

① 数値（整数型）の場合

```
;=================
; A%=USR(&H1234)
;=================
;
GETA:    MOV CX,[BX]  ; CX=&H1234
;
```

② 文字の場合

```
;
;====================
;   A$=USR("ABC")
;====================
;
GETP:    LDS BX,[BX]  ; DS:BX=ストリングディスクリプタ
         MOV CX,[BX]  ; CL=文字列の長さ（バイト数）=3
                        CH=リロケーションコード
                        (CH=0 のときDXがセグメント
                         CH<>0 のとき60Hがセグメント)
;
         CMP CH,0     ; IF CH=0 THEN SEGDX
         JE SEGDX
         XOR CH,CH    ; CH=0 : CX=文字列の長さ(3)
         MOV DX,60H   ; CH<>0 であったので60Hをセット
SEGDX:   MOV SI,2[BX] ; SI=文字列の先頭オフセットアドレス
         MOV DS,DX    ; DS < CH=0のときDX
                             CH<>0のとき60H
         MOV AL,[SI]  ; AL=第1文字目("A")
```

では次にサンプルプログラムを示します。数値では&H 1234を文字では"XYZ"を引数としてマシン語ルーチンに渡し，それらの引数をメモリに格納するものです。

引数&H1234をメモリに格納

```
1 ' save "SAMP3"
100 DEF SEG=&H1F00
110 DEF USR=&H0
120 A=USR(&H1234)
130 END
                         ;===================
                         ; [WORK] <= &H1234
                         ;===================
  0008                   WORK     EQU 0008H
0000 8B07                GETP:    MOV AX,[BX]
0002 0E                           PUSH CS
0003 1F                           POP DS
0004 A30800                       MOV .WORK,AX
0007 CF                           IRET
0008                     WORK     PW 1
```

引数XYZをメモリに格納

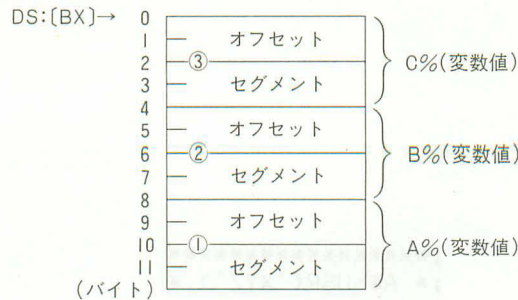
```
1 ' save "SAMP4"
100 DEF SEG=&H1F00
110 DEF USR=&H0
120 A$=USR("XYZ")
130 END
```

```
                              ;*****************
                              ;* A$=USR("XYZ") *
                              ;*****************
                              ;
0000 50                       ENTRY:  PUSH AX
0001 3C03                             CMP AL,03H
0003 7521           0026              JNE BACK
0005 C51F                             LDS BX,[BX]
0007 8B0F                             MOV CX,[BX]
0009 80FD00                           CMP CH,00H
000C 7405           0013              JE SEGDX
000E B500                             MOV CH,00H
0010 BA6000                           MOV DX,60H
                              ;
0013 8B7702                   SEGDX:  MOV SI,2[BX]
0016 8EDA                             MOV DS,DX
0018 0E                               PUSH CS
0019 07                               POP ES
001A BF2800                           MOV DI,OFFSET DATA
                              ;
001D 8A04                     GETCHR: MOV AL,[SI]
001F 268805                           MOV ES:[DI],AL
0022 46                               INC SI
0023 47                               INC DI
0024 E2F7           001D              LOOP GETCHR
                              ;
0026 58                       BACK:   POP AX
0027 CF                               IRET
                              ;
  0028                        ESG     EQU OFFSET $
                                      ESEG
                                      ORG ESG
0028                          DATA    RB 03H
                              ;
                                      END
```

10-2-3 CALL 文の引数

CALL 文が引数を持っている場合には，引数で指定された変数が置かれている番地のテーブルを参照して，引数を受け取るようになります。これを〝引数テーブル〟といい，そのセグメントはDS，オフセット（開始番地）は BX で示されます。

引数テーブルは，引数の値の入っている領域の開始番地のセグメントベースアドレスとオフセット値が組になっており，高位の番地から低位の番地に向けて順次格納されています。

これを図示しますと次のページのようになります。

CALL 文が USR 関数と異なり注意する点は次のとおりです。

- CALL 文が渡す引数の番地は，USR 関数の場合の FAC の番地ではなく，VARPTR 関数の値と同じ変数値の置かれている番地です。
- CALL 文では複数の引数が与えることができます。また，与えられた引数の番地を介して BASIC に値を返すことができます。
- 引数が文字型の場合には準備される番地はストリング・ディスクリプタの番地です。
- CALL 文は引数の個数や型については何ら情報を与えません。従って，これらを一致させることに注意しなければなりません。

以下，引数の受け取り方について数値（整数型）と文字を例にとり説明します（文字列の情報についてはUSR 関数の場合と同様です)。

引数が数値（整数型）の場合

① 数値が1個の場合

```
;==========================
; A%=&H1234:CALL SUB(A%)
;==========================
;
GETONE: LDS SI,[BX] ; DS:SI に引数のアドレス
        MOV AX,[SI] ; AX=&H1234
```

② 数値が2個の場合

```
;===========================
; A%=1:B%=2:CALL SUB(A%,B%)
;===========================
;
GETTWO: LES SI,[BX]      ; ES:SI に2番目の引数のアドレス
        MOV AX,ES:[SI]   ; AX=B%=2
        LES SI,4[BX]     ; ES:SI に1番目の引数のアドレス
        MOV BX,ES:[SI]   ; BX=A%=1
```

③　数値が３個の場合

```
;============================
; A%=1:B%=2:C%=3
; CALL SUB(A%,B%,C%)
;============================
;
GETTHR: LES SI,[BX]      ; ES:SI に3番目の引数のアドレス
        MOV AX,ES:[SI]   ; AX=C%=3
        LES SI,4[BX]     ; ES:SI に2番目の引数のアドレス
        MOV CX,ES:[SI]   ; CX=B%=2
        LES SI,8[BX]     ; ES:SI に1番目の引数のアドレス
        MOV DX,ES:[SI]   ; DX=A%=1
```

引数が文字列のとき

①　文字が１つの場合

```
;============================
; A$="ABC":CALL SUB(A$)
;============================
;
GETSTR: LDS BX,[BX]     ; DS:BX ストリングディスクリプタ
        MOV CX,[BX]     ; CL:文字列の長さ CH=セグメントコード・00のときDXがセグメント
        CMP CH,0        ; CHが0のときDXがセグメント             ・00以外ならば60Hがセグメント
        JE SEGMT        ; SEGMTにジャンプ
        XOR CH,CH       ; CH=0, CX=文字列の長さ
        MOV DX,60H      ; CH<>0 だったのでDX=60Hをセット
SEGMT:  MOV SI,2[BX]    ; SI=文字列の先頭オフセットアドレス
        MOV DS,DX       ; DS<CH=0のときDXの値
                        ;   CH<>0 のとき60H
        MOV AL,[SI]     ; AL=第1文字目
```

②　文字列が２つの場合

```
;============================
;    A$="ABC":B$="XYZ"
;    CALL SUB (A$,B$)
;============================
;
TWOSTR: LES SI,[BX]      ; ES:SIに2番目の引数のアドレス
        MOV CX,ES:[SI]   ; CL=文字列の長さ   CH=セグメントコード
        MOV SI,ES:2[SI]  ; SI=オフセット
        LES DI,4[BX]     ; ES:DIに1番目の引数のアドレス
        MOV AX,ES:[DI]   ; AL=文字列の長さ   AH=セグメントコード
        MOV DI,ES:2[DI]  ; DI=オフセット
```

というぐあいになります。これはレジスタを使いわけている方法です。次に同じレジスタを使っての具体例をあげます。

これは，２つの引数をメモリに格納するものです。

２つの引数をメモリに格納

```
1 ´save "abc
10 CLEAR,&H1F00:DEF SEG =&H1F00
20 A$="ABC"
30 B$="XYZ"
40 SUB=0
50 CALL SUB(A$,B$)
```

```
MON
h]C1F00
h]D3A
003A 58 59 5A 00 00 41 42 43 00 00 00 00 00 00 00 00    XYZ  ABC
h]^B
Ok
```

```
                            ;***************************
                            ;* A$="ABC":B$="XYZ"    *****
                            ;* CALL SUB(A$,B$)     ******
                            ;***************************
                            ;
0000 C437                   PARA2:  LES SI,[BX]       ; ES:SI SEGMENT & OFFSET
0002 268B0C                         MOV CX,ES:[SI]    ; CL=LENGTH,CH=SEGMENT CODE
0005 268B7402                       MOV SI,ES:2[SI]   ; OFFSET
0009 BF3A00                         MOV DI,ES:OFFSET DATA1
000C 1E                             PUSH DS           ; SAVE DS
000D E81200        0022             CALL STORE        ; PARA2=B$
0010 1F                             POP DS            ; RESTORE DS
                            ;
0011 C47704                 PARA1:  LES SI,4[BX]      ; ES:SI
0014 268B0C                         MOV CX,ES:[SI]    ; CL=LENGHT,CH=SEGMENT CODE
0017 268B7402                       MOV SI,ES:2[SI]   ; SI=OFFSET
001B BF3F00                         MOV DI,ES:OFFSET DATA2
001E E80100        0022             CALL STORE        ; PARA1=A$
                            ;
0021 CF                             IRET              ; BACK TO BASIC
                            ;
0022 0E                     STORE:  PUSH CS
0023 07                             POP ES            ; ES=CS
                            ;
0024 80FD00                         CMP CH,00
0027 7405          002E             JE SEGDX
0029 32ED                           XOR CH,CH
002B BA6000                         MOV DX,60H
002E 8EDA                   SEGDX:  MOV DS,DX
0030 8A04                   LP:     MOV AL,[SI]
0032 268805                         MOV ES:[DI],AL
0035 46                             INC SI
0036 47                             INC DI
0037 E2F7          0030             LOOP LP
0039 C3                             RET
                            ;
  003A                      DA      EQU OFFSET $
                                    ESEG
                                    ORG DA
                            ;
003A                        DATA1   RS 05H
003F                        DATA2   RS 05H
                            ;
                                    END
```

③ 数値・文字列混合（3つ）の場合

```
;===============================
; A$="ABC":B$="XYZ":C%=&H1234
;   CALL SUB(A$,B$,C%)
;===============================
        ORG 0
;
THREE:  LES SI,[BX]         ; ES:SIに3番目の引数のアドレス
        MOV AX,ES:[SI]      ; AX=C%=&H1234
        LES SI,4[BX]        ; ES:SIに2番目の引数のアドレス
        MOV CX,ES:[SI]      ; CL=文字列の長さ  CH=セグメントコード
        MOV SI,ES:2[SI]     ; SI=オフセット
        LES DI,8[BX]        ; ES=DIに1番目の引数のアドレス
        MOV BX,ES:[DI]      ; BL=文字列の長さ  BH=セグメントコード
        MOV DI,ES:2[DI]     ; DI=オフセット
```

では，これもメモリに格納する具体例をあげます。

数値・文字3つをメモリに格納

```
1 'save "abc2"
100 CLEAR,&H1F00:DEF SEG =&H1F00
110 A$="ABC"
120 B$="XYZ"
130 C%=&H1234
140 SUB=0
150  CALL SUB(A$,B$,C%)
160 END

MON
h]C1F00
h]D48
0048 [34 12] 00 00 00 [58 59 5A] 00 00 [41 42 43] 00 00 00        4    XYZ   ABC
h]^B   ↑                ↑              ↑
Ok    C%=&H1234       B$="XYZ"       A$="ABC"
```

```
                          ;********************************
                          ;* A$="ABC":B$="XYZ":C%=&H1234 *
                          ;* SUB=0:CALL SUB(A$,B$,C%)    *
                          ;********************************
                          ;
0000 C437                 PARA3:  LES SI,[BX]        ; ES:SI SEGMENT & OFFSET
0002 268B04                       MOV AX,ES:[SI]     ; AX=C%
0005 0E                           PUSH CS
0006 07                           POP ES             ; ES=CS
0007 BF4800                       MOV DI,ES:OFFSET DATA1
000A 268905                       MOV ES:[DI],AX     ; PARA3=C%
                          ;
000D C47704               PARA2:  LES SI,4[BX]       ; ES:SI SEGMENT & OFFSET
0010 268B0C                       MOV CX,ES:[SI]     ; CL=LENGTH,CH=SEGMENT CODE
0013 268B7402                     MOV SI,ES:2[SI]    ; OFFSET
0017 BF4D00                       MOV DI,ES:OFFSET DATA2
001A E81100         002E          CALL STORE         ; PARA2=B$
                          ;
001D C47708               PARA1:  LES SI,8[BX]       ; ES:SI
0020 268B0C                       MOV CX,ES:[SI]     ; CL=LENGHT,CH=SEGMENT CODE
0023 268B7402                     MOV SI,ES:2[SI]    ; SI=OFFSET
0027 BF5200                       MOV DI,ES:OFFSET DATA3
002A E80100         002E          CALL STORE         ; PARA1=A$
                          ;

002D CF                           IRET               ; BACK TO BASIC
                          ;
002E 1E                   STORE:  PUSH DS            ; SAVE DS
002F 0E                           PUSH CS
0030 07                           POP ES             ; ES=CS
                          ;
0031 80FD00                       CMP CH,00
0034 7405           003B          JE SEGDX
0036 32ED                         XOR CH,CH
0038 BA6000                       MOV DX,60H
003B 8EDA                 SEGDX:  MOV DS,DX
003D 8A04                 LP:     MOV AL,[SI]
003F 268805                       MOV ES:[DI],AL
0042 46                           INC SI
0043 47                           INC DI
0044 E2F7           003D          LOOP LP
0046 1F                           POP DS             ; RESTORE DS
0047 C3                           RET
                          ;
  0048                    DA      EQU OFFSET $
                                  ESEG
                                  ORG DA
                          ;
0048                      DATA1   RS 05H
004D                      DATA2   RS 05H
0052                      DATA3   RS 05H
                                  END
```

10-3　結果の戻し方

マシン語ルーチンに引数を渡した後，処理を行いBASICに結果を戻すにはどうすれば良いのでしょうか？答えは簡単です。引数の引き渡しに使ったレジスタを用いて，引数をもらったアドレスに格納すればOKです。以下，順にサンプルプログラムを示しながら説明していきます。

10-3-1 USR関数

① 数値（整数型）の場合

1234Hを引数としてそれを2倍にして戻します。

引数1234Hを2倍

```
1 'save "AX1234"
10 DEF SEG =&H1F00
20 DEF USR=&H0
30 A%=USR(&H1234) ←──引数 &H1 2 3 4
40 PRINT HEX$(A%)
50 END

run
2468
Ok
```

```
                          ;==========================
                          ; AX<=&H1234:AX=AX+AX
                          ;==========================
                                   ORG 0
                          ;
0000 50                   AX1234:  PUSH AX
0001 8B07                          MOV  AX,[BX] ; AX=1234H
0003 03C0                          ADD  AX,AX   ; AX=AX+AX
0005 8907                          MOV  [BX],AX ; AXの値を戻す
0007 58                            POP  AX
0008 CF                   BACK2:   IRET
```

② 文字の場合

文字列〝XYZ〟を引数として，それを小文字〝xyz〟にするものです。実行後の行番号30に注目して下さい。プログラム自身が変わっています。

引数XYZを小文字

```
1 'save "XYZ"
10 DEF SEG =&H1F00
20 DEF USR=&H0
30 A$=USR("XYZ") ←──引数"XYZ"
40 PRINT A$
50 END
```

```
run
xyz
Ok
list
1 ´save "XYZ"
10 DEF SEG =&H1F00
20 DEF USR=&H0
30 A$=USR("xyz")←——実行後，小文字に
40 PRINT A$           変わっている
50 END
Ok
```

```
                        ;*****************
                        ;* A$=USR("XYZ") *
                        ;*  ENTRY : XYZ  *
                        ;*  EXIT  : xyz  *
                        ;*****************
                        ;
0000 50                 SAVE:   PUSH AX          ; SAVE AX=TYPE
0001 C51F               GET:    LDS BX,[BX]      ; DS:BX STRING INFO
0003 8B0F                       MOV CX,[BX]      ; CL=LENGTH,CH=SEGMENT
0005 80FD00                     CMP CH,00H
0008 7405         000F          JE SEGDX
000A 32ED                       XOR CH,CH
000C BA6000                     MOV DX,60H       ; IF CH<>0 THEN DS=60H
000F 8B7702             SEGDX:  MOV SI,2[BX]
0012 8EDA                       MOV DS,DX
0014 8A04               STORE:  MOV AL,[SI]      ; AL=STRINGS
0016 0420               CONV:   ADD AL,20H       ; AL=SMALL LETTER
0018 8804                       MOV [SI],AL      ; STORE BACK
001A 46                         INC SI
001B E2F7         0014          LOOP STORE
001D 58                         POP AX           ; RESTORE AX
001E CF                         IRET             ; BACK TO BASIC
```

10-3-2 CALL文の場合

① 数値の場合

ここでは引数が2個のときを考えてみましょう。A％＝100，B％＝200としてコールし，マシン語ルーチンでA％とB％の値を入れ替えています。

2つの引数値を入れ替える

```
1 ´save "CALL2"
10 DEF SEG =&H1F00
20 SUB=0
30 A%=100
40 B%=200
50 CALL SUB(A%,B%)
60 PRINT A%,B%
70 END

run
 200           100
Ok
```

```
                          ;***********************
                          ;* CALL SUB(A%,B%)     *
                          ;* ENTRY: A%=100:B%=200 *
                          ;* EXIT : A%=200:B%=100 *
                          ;***********************
                          ;
0000 C437                 GET:   LES SI,[BX]        ; ES:SI=SECOND PARA
0002 268B04                      MOV AX,ES:[SI]     ; AX=B%=200
0005 C47F04                      LES DI,4[BX]       ; ES:DI=FIRST PARA
0008 268B1D                      MOV BX,ES:[DI]     ; BX=A%=100
                          ;
000B 268905                      MOV ES:[DI],AX     ; STORE AX AT BX AREA
000E 26891C                      MOV ES:[SI],BX     ; STORE BX AT AX AREA
                          ;
0011 CF                          IRET               ; BACK TO BASIC
```

② **文字の場合**

これも引数が2つのときを考えてみます。A＄＝〝ABC〞：B＄＝〝XYZ〞としてコールし，それぞれの真中の文字BとYとを小文字にしてみましょう。

2つの引数B・Yを小文字に

```
1 'save "CALL2"
10 DEF SEG=&H1F00
20 SUB=0
30 A$="ABC"
40 B$="XYZ"
50 CALL SUB(A$,B$)
60 PRINT A$,B$
70 END


run
AbC           XyZ
Ok
                          ;****************************
                          ;* CALL SUB(A$,B$)          *
                          ;* ENTRY: A$="ABC":B$="XYZ" *
                          ;* EXIT : A$="AbC":B$="XyZ" *
                          ;****************************
0000 C437                 PARA2: LES SI,[BX]        ; ES:SI=SEGMENT & OFFSET
0002 268B0C                      MOV CX,ES:[SI]     ; CL=LENGTH,CH=SEGMENT
0005 268B7402                    MOV SI,ES:2[SI]    ; SI=OFFSET
0009 E80E00        001A          CALL CHANGE
                          ;
000C C47704               PARA1: LES SI,4[BX]
000F 268B0C                      MOV CX,ES:[SI]
0012 268B7402                    MOV SI,ES:2[SI]
0016 E80100        001A          CALL CHANGE
                          ;
0019 CF                          IRET               ; BACK TO BASIC
                          ;
001A 1E                   CHANGE: PUSH DS           ; SAVE DS
001B 80FD00                      CMP CH,00H
001E 7405          0025          JE SEGDX
```

```
0020 32ED                  XOR CH,CH
0022 BA6000                MOV DX,60H
0025 8EDA          SEGDX:  MOV DS,DX
0027 46                    INC SI             ; SKIP TO MIDDLE
0028 8A04                  MOV AL,[SI]        ; GET MIDDLE LETTER
002A 0420                  ADD AL,20H         ; CONV TO SMALL LETTER
002C 8804                  MOV [SI],AL        ; STORE
002E 1F                    POP DS             ; RESTORE DS
002F C3                    RET                ; BACK TO MAIN
                   ;
                           END
```

③　POKE・PEEKを使う場合

USR関数やCALL文を使わなくて直接マシン語ルーチンに引数を受け渡したり，受け取ったりすることができます。それは，POKE・PEEKを使う方法で，マシン語ルーチンの決まったエリアに引数をPOKEして引き渡し，処理後，PEEKで結果を得るというものです。次にサンプルを示します。

255個以内のデータ数（N%）と実際のデータをPOKEした後，マシン語ルーチンでソートしBASICに戻った後，PEEKでソート結果を読み出しています。ちなみにこのソートはバブルソートと呼ばれるものです。

POKE・PEEKでの引数受け渡し

```
1 'save "buble"
100 DEF SEG=&H1F00
110 RANDOMIZE(VAL(RIGHT$(TIME$,2)))
120 INPUT "Data Number ";N%
130 POKE &H2F,N%
140 SUB=0
150 FOR I=&H30 TO &H30+N%-1
160 DT=INT(RND(1)*100)
170   POKE I,DT
180 NEXT
190 CALL SUB(N%) :PRINT
200 FOR I=&H30 TO &H30+N%-1
210  PRINT PEEK(I);" ";
220 NEXT
230 END

                   ;************************
                   ;* BUBBLE SORT
                   ;***********************
                   ;
0000 0E            START:  PUSH CS
0001 1F                    POP DS             ; DS=CS
                   ;
0002 BB2F00        SORT:   MOV BX,OFFSET DATA
0005 8A0F                  MOV CL,[BX]        ; CL=NO.OF DATA
0007 43                    INC BX
0008 8BF3          LOOP:   MOV SI,BX          ; [SI]=DATA OFFSET
000A 80E400                AND AH,00H         ; CLEAR AH=FLAG
```

```
0000D 8AE9                  MOV CH,CL       ; CH=COUNTER
000F FECD                   DEC CH
0011 8A04           NEXT:   MOV AL,[SI]     ; AL=FIRST
0013 8AF0                   MOV DH,AL
0015 8A5401                 MOV DL,01[SI]   ; DL=SECOND
0018 2AC2                   SUB AL,DL       ; IF AL<DL THEN SWAP
001A 7308      0024         JNB NOEX        ; ELSE NO EXCHANGE
001C 8814                   MOV [SI],DL
001E 887401                 MOV 01[SI],DH
0021 80CC01                 OR AH,01H
0024 46             NOEX:   INC SI
0025 FECD                   DEC CH
0027 75E8      0011         JNZ NEXT
0029 F6C401                 TEST AH,01H     ; SWAP?
002C 75DA      0008         JNE LOOP
                    ;
002E CF             RES:    IRET
                    ;
  002F              DA      EQU OFFSET $
                            DSEG
                            ORG DA
002F                DATA    RB 01H
0030                        RS 0FFH
                    ;
                            END
```

10-4 BASIC＋マシン語ルーチン

さて，この章のまとめとして，BASICとマシン語とをリンクしたルーチンをいくつか紹介します。

10-4-1 サウンドビープ

これはBEEP音を使っていろいろな音色を出すものです。RUNすると〝?〟が表示されますので16進数（0～FFFFH）を入力して下さい。

サウンドビープ

```
1 'save "SAMP5"
100 DEF SEG=&H1F00
110 DEF USR=&H0
120 DEFINT A-Z
130 INPUT DT$
140 DT=VAL("&H"+DT$)
150 A=USR(DT)
160 GOTO 120
```

```
                    ;******************
                    ;* SOUND BEEP   ****
                    ;******************
                    ;
0000 50             ENTRY:  PUSH AX
0001 8B17                   MOV DX,[BX]
0003 8AFE                   MOV BH,DH
0005 B105                   MOV CL,05H
```

```
0007 8ADA              LOOP1:  MOV BL,DL
0009 B006              BEEP1:  MOV AL,06H
000B E637                      OUT 37H,AL
000D FECB              ON:     DEC BL
000F 75FC        000D          JNE ON
0011 8ADA                      MOV BL,DL
0013 B007              BEEP0:  MOV AL,07H
0015 E637                      OUT 37H,AL
0017 FECB              OFF:    DEC BL
0019 75FC        0017          JNE OFF
001B FECF                      DEC BH
001D 75E8        0007          JNE LOOP1
001F FEC9                      DEC CL
0021 75E4        0007          JNE LOOP1
0023 58                BACK:   POP AX
0024 CF                        IRET
                       ;
                               END
```

10-4-2 小文字・大文字変換

英小文字の文字列を入力すると，大文字になって返ってきます。文字列以外のものや文字列でも英小文字以外はなにも処理をせずBASICに戻ります。

小文字・大文字変換

```
1 'save"Caps.bas"
10 DEF SEG=&H1F00
20 DEF USR=0
30 INPUT "String ";ST$
40 A$=USR(ST$)
50 PRINT "Caps --> ";ST$
60 PRINT :GOTO 30
```

```
                       ;*******************
                       ;* SMALL--> CAPS  **
                       ;*******************
                       ;
0000 50                ENTRY:  PUSH AX
0001 3C03                      CMP AL,03H
0003 7529        002E          JNE BACK
0005 C51F                      LDS BX,[BX]
0007 8B0F                      MOV CX,[BX]
0009 80F900                    CMP CL,00H
000C 7420        002E          JE BACK
000E 80FD00                    CMP CH,00H
0011 7405        0018          JE SEGDX
0013 B500                      MOV CH,00H
0015 BA6000                    MOV DX,60H
                       ;
0018 8B7702            SEGDX:  MOV SI,02[BX]
001B 8EDA                      MOV DS,DX
001D 8A04              GETCHR: MOV AL,[SI]
001F 3C61                      CMP AL,61H
0021 7208        002B          JB NEXT
0023 3C7A                      CMP AL,7AH
```

```
0025 7704         002B           JA NEXT
0027 2C20                        SUB AL,20H
0029 8804                        MOV [SI],AL
                          ;
002B 46                   NEXT:  INC SI
002C E2EF         001D           LOOP GETCHR
                          ;
002E 58                   BACK:  POP AX
002F CF                          IRET
                          ;
                                 END
```

10-4-3 最大値を求める

まず，データ数を入力すると，データをランダムに作ります。そして，そのデータを画面に表示するとともにメモリに書き込みます。その中で最大の数をサーチしてBASICに返します。

最大値サーチ

```
1 'save "SAMP9"
100 RANDOMIZE(VAL(RIGHT$(TIME$,2)))
110 INPUT "Data Number ";N%
120 DEF SEG=&H1F00
130 SUB=0
140 FOR I=&H23 TO &H23+N%-1
150 DT=INT(RND(1)*100):PRINT DT;" ";
160   POKE I,DT
170 NEXT
180 CALL SUB(N%) :PRINT
190 PRINT "Largest ";N%
200 END
```

```
                          ;*************************
                          ;* FIND LARGEST NUMBER   *
                          ;*************************
                          ;
0000 C537                 GETPAR: LDS SI,[BX]
0002 8B0C                         MOV CX,[SI]
0004 B000                         MOV AL,0
0006 1E                           PUSH DS
0007 0E                           PUSH CS
0008 07                           POP ES
0009 BB2200                       MOV BX,OFFSET WORK
000C BF2300                       MOV DI,OFFSET DATA
                          ;
000F 263805               COMP:   CMP ES:[DI],AL
0012 7205         0019            JB NOSWAP
0014 268605                       XCHG ES:[DI],AL
0017 8807                         MOV [BX],AL
0019 47                   NOSWAP: INC DI
001A E2F3         000F            LOOP COMP
                          ;
001C 8A07                         MOV AL,[BX]
001E 1F                           POP DS
001F 8804                         MOV [SI],AL
```

```
0021 CF                         IRET
                          ;
  0022                    ESG   EQU OFFSET $
                                ESEG
                                ORG ESG
0022                      WORK  RB 01H
0023                      DATA  RB 0FFH
                          ;
                                END
```

10-4-4 文字列を逆に表示

今度は，引数が文字列の場合で，ある文字列をマシン語ルーチンに渡すと，それを逆に表示するというものです。文字列を逆にプリントしてもあまり意味がないと思われる読者もいらっしゃるかと思いますが，これは言葉の遊びの1つと考えてください。ある文を逆から読んでも同じなのを回文（かいぶん）と言いますが，この回文を見つけるのに役立つことでしょう。回文の例として，タケヤブヤケタなどがあります。英文で有名なのが，Madam I'm Adam. で続けて読むとちゃんと同じになります。では，この回文作成プログラムを紹介しましょう

文字列を逆に表示

```
1 'save "KAI.BAS"
100 DEF SEG=&H1F00
110 SUB=0
120 INPUT "String ";ST$
130 CALL SUB(ST$)
140 PRINT "Kaibun = ";ST$
150 END
```

```
                          ;********************
                          ;* KAIBUN PRINT *****
                          ;********************
                          ;
0000 C51F                 ENTRY:  LDS BX,[BX]
0002 8B0F                         MOV CX,[BX]
0004 80FD00                       CMP CH,00H
0007 7405          000E           JE SEGDX
0009 B500                         MOV CH,00H
000B BA6000                       MOV DX,60H
                          ;
000E 8B7702               SEGDX:  MOV SI,02[BX]
0011 8EDA                         MOV DS,DX
0013 0E                           PUSH CS
0014 07                           POP ES
0015 BF3400                       MOV DI,OFFSET BUFF
                          ;
0018 33DB                         XOR BX,BX
001A 03D9                         ADD BX,CX
001C 4B                           DEC BX
001D 51                           PUSH CX
001E 57                           PUSH DI
001F 8A00                 GETCHR: MOV AL,[BX+SI]
```

```
0021 268805                 MOV ES:[DI],AL
0024 4B                     DEC BX
0025 47                     INC DI
0026 E2F7        001F       LOOP GETCHR
                  ;
0028 5F           PICKUP:   POP DI
0029 59                     POP CX
002A 268A05       STORE:    MOV AL,ES:[DI]
002D 8804                   MOV [SI],AL
002F 47                     INC DI
0030 46                     INC SI
0031 E2F7        002A       LOOP STORE
0033 CF                     IRET
                  ;
 0034             ESG       EQU OFFSET $
                            ESEG
                            ORG ESG
0034              BUFF      RB 0FFH
                  ;
                            END
```

10-4-5 ROLL 200 ＆ ROLL 400

ROLL 200 や ROLL 400 やそれ以上のロールアップができるのです。引数は通常のロールアップの指定に 80 を掛けたものにして下さい。例えば，ROLL 200 とするには，DT%＝200 ＊ 80 として CALL して下さい。引数を 80 の倍数以外にすると斜めにスクロールします。ROLL 400 を実行するには，ソースリストの GVRAM 4000 H と GEND 3 E 80 H に対応するところをそれぞれ 8000 H, 7 D 00 H に変えて下さい。

```
1 'save "RO"
10 WIDTH 80,25:SCREEN 0,0
20 LINE (0,0)-(639,199),1,BF
30 PUT(0,180),KANJI(&H3441)
40 DEF SEG=&H1F00
50 RL=0
60 D%=200*80
65 INPUT A
70 CALL RL(D%)
80 END

                ;*********************
                ;* ROLL 200:ROLL 400 *
                ;*********************
 4000           GVRAM    EQU 4000H        ; 16K BYTES. 8000H FOR ROLL 400
 3E80           GEND     EQU 3E80H        ; GVRAM END ADD. 3E80H*2 <= 400
                ;
0000 C437       GETP:    LES SI,[BX]
0002 268B34              MOV SI,ES:[SI]   ; GET PARA FROM BASIC
0005 B90040              MOV CX,GVRAM     ; CX=16K
0008 2BCE                SUB CX,SI        ; CX=NO.OF BYTES TO ROLL
000A 33FF                XOR DI,DI        ; DI=GVRAM OFFSET START
                ;
```

```
000C B800A8          BLUE:   MOV AX,0A800H    ; ROLL BLUE
000F E80D00    001F          CALL ROLL
                     ;
0012 B800B0          RED:    MOV AX,0B000H    ; ROLL RED
0015 E80700    001F          CALL ROLL
                     ;
0018 B800B8          GREEN:  MOV AX,0B800H    ; ROLL GREEN
001B E80100    001F          CALL ROLL
                     ;
001E CF                      IRET             ; BACK TO BASIC
                     ;
001F 56              ROLL:   PUSH SI          ; SAVE REGS
0020 57                      PUSH DI
0021 51                      PUSH CX
0022 56                      PUSH SI
0023 8ED8                    MOV DS,AX        ; ROLL SUBROUTINE
0025 8EC0                    MOV ES,AX        ; ES=DS=GVRAM
0027 FC                      CLD              ; INCREMENT
0028 F3A4                    REP MOVSB        ; MOVE BYTES
                     ;
002A B00C            DOFF:   MOV AL,0CH       ; DISP OFF
002C E6A2                    OUT 0A2H,AL
002E 59                      POP CX           ; CX=SI=NO.OF BYTES
002F 33C0                    XOR AX,AX        ; AX=0
0031 BF803E                  MOV DI,GEND      ; GVRAM OFFSET END
0034 FD                      STD              ; DECREMENT
0035 F3AA                    REP STOSB        ; CLEAR THE REST
                     ;
0037 B00D            DON:    MOV AL,0DH       ; DISP ON
0039 E6A2                    OUT 0A2H,AL
003B 59                      POP CX           ; RESTORE REGS
003C 5F                      POP DI
003D 5E                      POP SI
003E C3                      RET
                     ;
                             END
```

10-4-6 アドレスサーチ

これは，PC-9801内部の文字列をサーチしてそのアドレスを出力するものです。引数としてサーチする文字列（3～5バイト）とセグメントを渡すと，その文字列が格納されているオフセットアドレスをBASICに返します。文字列が見つからない場合は引数を0にしてBASICに戻しています。

アドレスサーチ① BASIC

```
1 'save "ADDSER.BAS"
100 CLS :A$="====  Address Search for PC-9801  ===="
110 PRINT A$ : DEF SEG=&H1F00 : KS=0
120 INPUT "Key Word ";KW$ : KLN=LEN(KW$)
130  POKE &H72,KLN
140 INPUT "Segment  ";SG$ :CLS
150  SG%=VAL("&H"+SG$) : A=1
160 FOR I=&H73 TO &H73+KLN-1
```

```
170  POKE I,ASC(MID$(KW$,A,1))
180  A=A+1
190 NEXT I
200  CALL KS(SG%)
210 AD%=SG% : IF AD%=0 THEN BEEP:PRINT "Not Found.":GOTO *ED
220 PRINT A$
230 PRINT "Key Word : ";KW$
240 PRINT "Segment  : ";SG$
250 PRINT "Offset   : ";RIGHT$("0000"+HEX$(AD%),4)
260 *ED
270 END
```

アドレスサーチ② マシン語

```
                    ;*****************
                    ;* ADDRESS SEARCH  *
                    ;* CALL KS(SG%)    *
                    ;* SG%<=OFFSET     *
                    ;*****************
                    ;
0000 8B4F02         START:  MOV CX,2[BX]      ; Get parameter
0003 8EC1                   MOV ES,CX         ; from BASIC CALL
0005 8B37                   MOV SI,0[BX]
0007 268B04                 MOV AX,ES:[SI]    ; AX=Segment to search
000A 06                     PUSH ES           ; Save ES and SI
000B 56                     PUSH SI           ; to return parameter to BASIC
000C 8EC0                   MOV ES,AX         ; Eseg=Segment to search
000E 0E                     PUSH CS
000F 1F                     POP DS            ; Data Seg=Code Seg
0010 B9FFFF                 MOV CX,0FFFFH     ; CX=No.of bytes (64k) to search

0013 BB7200                 MOV BX,OFFSET NWORD    ; [BX]=No.of keyword
0016 BE7300                 MOV SI,OFFSET KWORD    ; SI=Keyword address
0019 33FF                   XOR DI,DI         ; DI=0 Offset address found
                    ;
001B 8A04           BCOMP:  MOV AL,[SI]       ; AL=First Source byte
001D 263805         OUTLP:  CMP ES:[DI],AL    ; Compare First bytes
0020 7408      002A FD0:    JZ FD1            ; If equal then Found1
                    ;

0022 47                     INC DI
0023 49                     DEC CX
0024 75F7      001D         JNZ OUTLP
0026 33C0                   XOR AX,AX         ; If not found,then AX=0
0028 EB42      006C         JMPS BACK         ; Go back to BASIC
                    ;
002A 8A4401         FD1:    MOV AL,01H[SI]    ; AL=Second Source byte
002D 26384501               CMP ES:01H[DI],AL
0031 7404      0037         JZ FD2
0033 47                     INC DI
0034 49                     DEC CX
0035 EBE4      001B         JMPS BCOMP        ; If not equal, then start over
                    ;
0037 8A4402         FD2:    MOV AL,02H[SI]    ; AL=Third Source byte
003A 26384502               CMP ES:02H[DI],AL
003E 7404      0044         JZ FD3
0040 47                     INC DI
0041 49                     DEC CX
0042 EBD7      001B         JMPS BCOMP
```

```
                        ;
0044 8A17           FD3:      MOV DL,[BX]
0046 80FA03                   CMP DL,03H        ; If DL=3 then Back
0049 741F     006A            JE FD5
004B 8A4403                   MOV AL,03H[SI]
004E 26384503                 CMP ES:03H[DI],AL
0052 7404     0058            JZ FD4
0054 47                       INC DI
0055 49                       DEC CX
0056 EBC3           001B          JMPS BCOMP
                          ;
0058 80FA04               FD4:    CMP DL,04H        ; If DL=4 then Back
005B 740D           006A          JE FD5
005D 8A4404                       MOV AL,04H[SI]    ; AL=Fifth Source byte
0060 26384504                     CMP ES:04H[DI],AL
0064 7404           006A          JZ FD5
0066 47                           INC DI
0067 49                           DEC CX
0068 EBB1           001B          JMPS BCOMP
                          ;
006A 8BC7                 FD5:    MOV AX,DI         ; AX=DI=Offset address
006C 5E                   BACK:   POP SI            ; Restore SI & ES
006D 07                           POP ES
006E 268904                       MOV ES:[SI],AX    ; Return AX to BASIC
0071 CF                           IRET              ; Back to BASIC
                          ;
  0072                    DATA    EQU OFFSET $
                                  DSEG
                                  ORG DATA
0072                      NWORD   RS 01H
0073                      KWORD   RS 05H
                          ;
                          END
```

第 11 章　入出力ファイル

11-1　入出力装置とファイル
11-2　変数でファイル指定
11-3　ファイルバッファ
11-4　ファイルバッファ使用例
11-5　高速グラフィックス・ローダー

第11章 入出力ファイル

11-1 入出力装置とファイル

N_{88}-BASIC (86) は，本体と入出力装置との情報のやりとりを〝ファイル〟という概念で行っています。そのファイルの指定は，入出力装置を指定する〝デバイス名〟と〝ファイル名〟で行います。次に各入出力装置につけられているデバイス名の一覧を表11-1にあげます。

デバイス番号	デバイス名	入出力装置名	入力	出力
05H	KYBD:	キーボード	○	×
04H	SCRn:	スクリーン	×	○
B2H	LPT1:	プリンタ	×	○
1AH	CAS1:	カセットテープ (1200ボー)	○	○
	CAS2:	(600ボー)	○	○
B0H	1:	ディスク 1	○	○
	2:	2	○	○
	3:	3	○	○
	4:	4	○	○
	5:	5	○	○
	6:	6	○	○
	7:	7	○	○
	8:	8	○	○
	9:	9	○	○
	10:	10	○	○
19H	COM:	RS-232C ポート	○	○

表11-1 デバイス名の一覧

11-2 変数でファイル指定

デバイス名とファイル名をあわせてファイルディスクリプタと呼ばれます。これは文字変数で表わせるために，入出力装置の変更が簡単に行えます。

```
F$="2:DEMO"
OPEN F$ FOR OUTPUT AS #1
```

とすればディスクのドライブ2に書き込みが可能です。これをF$=〝SCRN:〟とすればCRT

に，F$＝〝LPT 1：〟とすればプリンタに出力先が変わります。

次にキーボードから入力したものをCRT，プリンタ，ディスクおよびカセットに出力する簡単なサンプルプログラムを示します。

デバイス変更プログラム

```
1 ´save "DEVICE.bas"
100 ´ File I/O sample
110 OPEN "KYBD:" FOR INPUT AS #1
120  INPUT #1,A$
130  F$="SCRN:" : GOSUB *FOUT ´ --- CRT
140  F$="LPT1:" : GOSUB *FOUT ´ --- PRINTER
150  F$="DEMO"  : GOSUB *FOUT ´ --- DISK DRIVE 1
160  F$="CAS1:" : GOSUB *FOUT ´ --- CASSETTE
170 CLOSE #1
180 END
190 ´ --- File Sub ---
200 *FOUT
210 OPEN F$ FOR OUTPUT AS #2
220  PRINT #2,A$
230 CLOSE #2
240 RETURN
```

11-3　ファイルバッファ

入出力装置とのやりとりは，ファイルバッファ（窓口）を通じて行われます。このバッファは16個用意されており，1個256バイトとなっています。N_{88}-BASIC（86）やN_{88}-Disk BASIC（86）を起動した後，

How many files （0—15）?

の問いに同時にオープンするファイルの数が指定できます。通常ユーザーが使うバッファは1～15までです。バッファ0はDSKI$やDSKO$やFILESを実行したときに，システムが使用します。次にファイルバッファがメモリマップ上でどの位置にあるかを，図11-3に示します。また，関連するワークエリア（ファイルコントロールブロック）などの位置もあわせて説明します。

アドレス	内容	サイズ
	ファイルバッファ（入出力バッファ）#0～#15 （256×（ファイル同時オープン数）＋1）バイト	256～4,096バイト
VARPTR（#n）＋32	物理I/Oコントロール・ブロック（FCB） （40×（ファイル同時オープン数＋1））バイト	
VARPTR（#n）	物理I/Oコントロール・ブロック （24×装置タイプ数）バイト	24～72バイト
	FAT用バッファ フロッピー…… 256×デバイス数 5Mハード……1,280×デバイス数　バイト 10Mハード……2,560×デバイス数	256～10,496バイト
	デバイスコントロール・ブロック（DCB） （20×デバイス数）バイト	20～200バイト
1D90H	媒体諸情報 （DSKF関数で得られるもの）	144バイト
1D00H セグメント60H		

ファイルコントロール・ブロック　（FCB）　は次のようになっています。

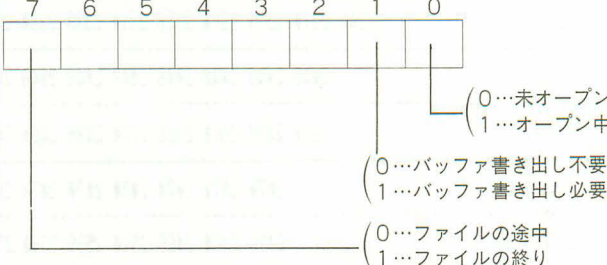
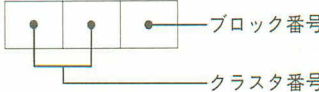
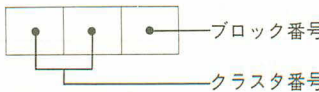
アドレス（16進）	バイト数	フィールド	説　　明
0	1	ファイル番号	#0～#15（00H～0FH）
1	1	オープンモード	40H……FOR OUTPUT 41H……FOR APPEND 80H……FOR INPUT C0H……ランダムアクセス
2	2	DCBアドレス	ファイルの属するDCBの先頭アドレス
4	2	次のFCBアドレス	同一デバイスに属する次のFCBアドレス
6	6	ファイル名	ファイル名（6文字に未たないときは20H）
C	3	拡張子	ファイル名の拡張子（ない場合は20H）
F	1	アトリビュート	ファイルの属性を示す ビット 7 6 5 4 3 2 1 0 ビット0：マシン語ファイル ビット4：0…書き込み可能／1…書き込み禁止 ビット5：1…Pオプション ビット6：0…ライトオンリー／1…リードアフターライト ビット7：0…アスキー形式／1…非アスキー形式

10	2	第１クラスタ	ファイル内の先頭クラスタアドレス
12	1	アトリビュート ワーク	ファイルの属性を示すワークエリア このフィールドにディレクトリ部が取り込まれ，以後，ここで属性のチェックが行われる。 フィールドの意味はアトリビュートと同じ
13	1	デバイスタイプ	ファイル番号が対応する装置を示す B0H……ディスク B2H……プリンタ 1AH……カセット 19H ……COM 04H ……CRT 05H ……キーボード
14	1	ファイル ステータス	ファイルの処理状態を示す 7 6 5 4 3 2 1 0 ビット0：0…未オープン／1…オープン中 ビット1：0…バッファ書き出し不要／1…バッファ書き出し必要 ビット7：0…ファイルの途中／1…ファイルの終り
15	3	データエンド アドレス	最終レコードアドレスを示す ブロック番号 クラスタ番号
18	2	レコード・エンド	最終レコード番号を示す
1A	3	ネクスト レコードアドレス	次のレコードアドレスを示す ブロック番号 クラスタ番号
1D	2	ネクスト レコードNo.	次のレコード番号を示す
1F	1	リザーブ	システムの予約部分
20	2	バッファアドレス	ファイルバッファのアドレスを示す
22	2	データポインタ	シーケンシャルファイルの時，次に読み書きするデータのアドレス

図11-3 ファイルバッファとワークエリア

次に，Disk BASIC 起動時のファイルバッファのアドレス一覧表を示します。
これはROM およびDisk のバージョンにより異なることがあります。

ファイルバッファアドレス一覧表(DiSK BASIC)

# OF OPEN FILES		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
# 0	F C B TOP	1FD0	1FD0	1FD0	1FD0	1FD0	1FD0	1FD0	1FD0	1FD0	1FD0	1FD0	1FD0	1FD0	1FD0	1FD0	1FD0
	BUFFER TOP	1FF8	2020	2048	2070	2098	20C0	20E8	2110	2138	2160	2188	21B0	21D8	2200	2228	2250
# 1	F C B TOP		1FF8	1FF8	1FF8	1FF8	1FF8	1FF8	1FF8	1FF8	1FF8	1FF8	1FF8	1FF8	1FF8	1FF8	1FF8
	BUFFER TOP		2120	2148	2170	2198	21C0	21E8	2210	2238	2260	2288	22B0	22D8	2300	2328	2350
# 2	F C B TOP			2020	2020	2020	2020	2020	2020	2020	2020	2020	2020	2020	2020	2020	2020
	BUFFER TOP			2248	2270	2298	22C0	22E8	2310	2338	2360	2388	23B0	23D8	2400	2428	2450
# 3	F C B TOP				2048	2048	2048	2048	2048	2048	2048	2048	2048	2048	2048	2048	2048
	BUFFER TOP				2370	2398	23C0	23E8	2410	2438	2460	2488	24B0	24D8	2500	2528	2550
# 4	F C B TOP					2070	2070	2070	2070	2070	2070	2070	2070	2070	2070	2070	2070
	BUFFER TOP					2498	24C0	24E8	2510	2538	2560	2588	25B0	25D8	2600	2628	2650
# 5	F C B TOP						2098	2098	2098	2098	2098	2098	2098	2098	2098	2098	2098
	BUFFER TOP						25C0	25E8	2610	2638	2660	2688	26B0	26D8	2700	2728	2750
# 6	F C B TOP							20C0	20C0	20C0	20C0	20C0	20C0	20C0	20C0	20C0	20C0
	BUFFER TOP							26E8	2710	2738	2760	2788	27B0	27D8	2800	2828	2850
# 7	F C B TOP								20E8	20E8	20E8	20E8	20E8	20E8	20E8	20E8	20E8
	BUFFER TOP								2810	2838	2860	2888	28B0	28D8	2900	2928	2950
# 8	F C B TOP									2110	2110	2110	2110	2110	2110	2110	2110
	BUFFER TOP									2938	2960	2988	29B0	29D8	2A00	2A28	2A50
# 9	F C B TOP										2138	2138	2138	2138	2138	2138	2138
	BUFFER TOP										2A60	2A88	2AB0	2AD8	2B00	2B28	2B50
#10	F C B TOP											2160	2160	2160	2160	2160	2160
	BUFFER TOP											2B88	2BB0	2BD8	2C00	2C28	2C50
#11	F C B TOP												2188	2188	2188	2188	2188
	BUFFER TOP												2CB0	2CD8	2D00	2D28	2D50

```
      F C B  TOP                                                        21B0 21B0 21B0 21B0
#12  -----------------------------------------------------------------------------------------
      BUFFER TOP                                                        2D08 2E00 2E28 2E50
---------------------------------------------------------------------------------------------
      F C B  TOP                                                             21D8 21D8 21D8
#13  -----------------------------------------------------------------------------------------
      BUFFER TOP                                                             2F00 2F28 2F50
---------------------------------------------------------------------------------------------
      F C B  TOP                                                                  2200 2200
#14  -----------------------------------------------------------------------------------------
      BUFFER TOP                                                                  3028 3050
---------------------------------------------------------------------------------------------
      F C B  TOP                                                                       2228
#15  -----------------------------------------------------------------------------------------
      BUFFER TOP                                                                       3150
---------------------------------------------------------------------------------------------
      TEXT   TOP  20F8 2220 2348 2470 2598 26C0 27E8 2910 2A38 2B60 2C88 2DB0 2ED8 3000 3128 3250
---------------------------------------------------------------------------------------------
```

ちなみに，ファイルバッファアドレス出力プログラムもあわせて紹介しておきます。

ROM BASIC のときは，130 行の FCB. TOP を &H1D00 として下さい。なお，これは日本語 BASIC が起動されているとアドレスが異なります。

```
0 'SAVE"FCB.PRT"
100  OPEN "SCRN:" FOR OUTPUT AS 1
110  'PRINT #1,CHR$(27)"Q";:REM PC-8023
120  'PRINT #1,CHR$(15);   :REM MP-82 TYPE-II
130  FCB.TOP=&H1FD0:REM ROM 1D00H
140  MAX.FILE=15
150 '
160  PRINT #1," # OF OPEN FILES ";
170   FOR I=0 TO MAX.FILE
180    PRINT #1,USING "  ## ";I;
190   NEXT
200   PRINT #1,""
210  REAL.LINE=(MAX.FILE+1)*5+18
220  DOT.LINE=REAL.LINE-6
230  PRINT #1,STRING$(REAL.LINE,"-")
240 '
250'FOR J=0 TO 15
260  PRINT #1,"      F C B   TOP   ";
270   FOR I=0 TO MAX.FILE
280    IF I<J THEN PRINT #1,"     ";:GOTO *SKIP.FCB
290    PRINT #1,HEX$(FCB.TOP+J*&H28)" ";
300 *SKIP.FCB
310   NEXT
320    PRINT #1,""
330  PRINT #1," #";:PRINT #1,USING"##  ";J;
335  PRINT #1,STRING$(DOT.LINE,"-")
340  PRINT #1,"      BUFFER TOP   ";
350   FOR I=0 TO MAX.FILE
360    BUF.TOP=FCB.TOP+(I+1)*&H28
370     IF I<J THEN PRINT #1,"     ";:GOTO *SKIP.BUF
380      PRINT #1,HEX$(BUF.TOP+&H100*J)" ";
390 *SKIP.BUF
400   NEXT
410   PRINT #1,""
```

```
420   PRINT #1,STRING$(REAL.LINE,"-")
430 NEXT J
440 '
450  PRINT #1,"     TEXT    TOP   ";
460   FOR I=0 TO MAX.FILE
470    TXT.TOP=FCB.TOP+(I+1)*&H128
480     PRINT #1,HEX$(TXT.TOP)" ";
490   NEXT
500  PRINT #1,""
510  PRINT #1,STRING$(REAL.LINE,"-")
520 '
```

11-4　ファイルバッファ使用例

それでは，実際にファイルバッファおよびFCBにどのように書き込まれていくか見てみましょう。

まず次のプログラムを実行して下さい。なお，ここでは8インチのディスクユニットを例にとっていますので，ミニをお使いの方や8インチとミニを両方起動している場合にはアドレスが異なります。また，データが書き込まれる先頭クラスタもディスケットにより違ってきます。

```
10 OPEN "TEST" FOR OUTPUT AS #1
20  SG=VARPTR(#1,1)
30  OF=VARPTR(#1)
40 PRINT HEX$(SG),HEX$(OF)
50 PRINT HEX$(OF+&H20) ' File buffer
```

```
run
60←FCBセグメント 1FF8←FCBオフセット
2018←ファイルバッファ・オフセット
Ok
MON
hJC60
hJD1FF8
1FF8 01 40 90 1D 00 00 54 45 53 54 20 20 20 20 20 00      @┴   TEST
hJ
2008 92 00 00 B0 81 92 00 00 00 00 92 00 01 01 00 02   ┤ ─▁┤    ┤
hJ
2018 48 21 00 00 00 00 00 00 02 00 00 00 00 00 00 00    H!
hJD2148
2148 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
hJ^B
Ok
```

（図中注記：01＝ファイル番号，40＝オープンモード，90 1D＝DCB，00 00＝FCB，54 45 53 54 20 20 20 20 20＝ファイル名，00＝アトリビュート；92 00＝先頭クラスタ，B0＝デバイスタイプ，81＝ステータス，92 00 00＝最終レコードアドレス，00 00＝レコードNo.，92 00 01＝次のレコードアドレス，01 00＝次のレコードNo.；48 21＝ファイルバッファ；ファイルバッファはすべて00）

次にデータを書き込みます。

```
PRINT #1,"ABCDEF";
Ok
MON
h]C60
h]D1FF8
1FF8 01 40 90 1D 00 00 54 45 53 54 20 20 20 20 20 00        @┴   TEST
h]
2008 92 00 00 B0 83 92 00 00 00 00 92 00 01 01 00 00        ┤  ─▬┤     ┤
h]
2018 48 21 [06] 00 00 00 06 00 02 00 00 00 00 00 00 00       H!
h]D2148      └ データポインタ（6文字のデータ）
2148 41 42 43 44 45 46 00 00 00 00 00 00 00 00 00 00        ABCDEF
h]^B   データがファイルバッファに格納
Ok
```

さらに続けてデータを書きます。

このときには，RETURN を押したことになります。

```
PRINT #1,"XYZ"
Ok
MON
h]C60
h]D1FF8
1FF8 01 40 90 1D 00 00 54 45 53 54 20 20 20 20 20 00        @┴   TEST
h]
2008 92 00 00 B0 83 92 00 00 00 00 92 00 01 01 00 00        ┤  ─▬┤     ┤
h]
2018 48 21 [0B] 00 00 00 00 00 02 00 00 00 00 00 00 00       H!
h]D2148      └ データポインタ（11文字のデータ）
2148 41 42 43 44 45 46 58 59 5A [0D] [0A] 00 00 00 00 00      ABCDEFXYZ
h]^B          データ                 CR   LF
Ok
```

ここで長い文字列を書き込んでみましょう。するとディスクをアクセスしました。

```
PRINT #1,STRING$(250,"@")
Ok
MON
h]C60
h]D1FF8
1FF8 01 40 90 1D 00 00 54 45 53 54 20 20 20 20 20 00     @┴   TEST
h]                                  レコード No.      次のレコード No.
2008 92 00 00 B0 83 92 00 01 [01 00] 92 00 02 [02 00] 00     ┤  ─▬┤     ┤
h]
2018 48 21 07 00 00 00 00 00 02 00 00 00 00 00 00 00      H!
h]D2148
2148 40 40 40 40 40 [0D] [0A] 59 5A 0D 0A 40 40 40 40 40     @@@@@  YZ  @@@@@
h]        レコード      CR   LF   レコードの残り
```

```
2158 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40    @@@@@@@@@@@@@@@@
h]
2168 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40    @@@@@@@@@@@@@@@@
h]^B
Ok
```

バッファが一杯になったため，ディスクに書き込んでいます。また，レコードNo.が1つ増えています。

最後にCLOSEします。ディスクにアクセスしました。

```
CLOSE #1
Ok
MON
h]C60                    ファイル番号とデバイスタイプだけが残りすべてクリア
h]D1FF8
1FF8 [01] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
h]
2008 00 00 00 [B0] 00 00 00 00 00 00 00 00 00 00 00 00
h]
2018 48 21 00 00 00 00 00 00 02 00 00 00 00 00 00 00      H!
h]D2148
2148 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
h]
2158 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
h]^B
Ok                   ファイルバッファもすべてクリア
```

例では，8インチディスクの92Hクラスタに書き込みました。92Hクラスタは，トラック49H，サーフェス0，セクタ1となりますので，モニタのダンプコマンドで書き込んだ内容を確認してみましょう。モニターモードで次のコマンドを実行して下さい。

[CTRL] D1，0，49，1，49，2

```
Dr 01,Sur 00,Tr 0049,Sec 01          CR   LF
0000 41 42 43 44 45 46 58 59 5A [0D] [0A] 40 40 40 40 40    ABCDEFXYZ  @@@@@
0010 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40    @@@@@@@@@@@@@@@@
0020 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40    @@@@@@@@@@@@@@@@
0030 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40    @@@@@@@@@@@@@@@@
0040 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40    @@@@@@@@@@@@@@@@
0050 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40    @@@@@@@@@@@@@@@@
0060 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40    @@@@@@@@@@@@@@@@
0070 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40    @@@@@@@@@@@@@@@@
0080 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40    @@@@@@@@@@@@@@@@
0090 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40    @@@@@@@@@@@@@@@@
00A0 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40    @@@@@@@@@@@@@@@@
00B0 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40    @@@@@@@@@@@@@@@@
00C0 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40    @@@@@@@@@@@@@@@@
00D0 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40    @@@@@@@@@@@@@@@@
```

```
00E0 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40   @@@@@@@@@@@@@@@@
00F0 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40   @@@@@@@@@@@@@@@@
Dr 01,Sur 00,Tr 0049,Sec 02
0000 40 40 40 40 40 0D 0A 1A 00 00 00 00 00 00 00 00   @@@@@
0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

データの終りを示します。

11-5 高速グラフィックス・ローダー

この章のまとめとして１つの課題に挑戦してみましょう。その課題とは，PC-8801の640×200モードのカラーグラフィックスをデータとしてディスクに書き込んでおき，PC-9801上に持って来ることです。つまり，PC-8801で作られたいろいろなカラーグラフィックスをPC-9801で利用しようという訳です。

PC-8801のグラフィックスは次のようにランダムファイルとして書き込まれているものとします。

PC-8801のグラフィックVRAMの青，赤，緑の画面それぞれ16KをPUTしていることになります。PC-8801ではBASICで直接グラフィックVRAMへのアクセスはできませんので，マシン語でバンク切り換えした後，ランダムファイルバッファに送り込んで，PUTしています。ちなみにそのプログラムを示します。

PC-8801用 G-VRAMセーブ

```
1 ´save "SAVE88.RAM"
100 *SAVESUB  ´ for PC-8801
110 F$="GVRAM.88"
120 DEF USR=&HBF00
130 DEF USR1=&HBF20 : DEF USR2=&HBF40
140 ´========= Write Macine Code =========
150 RESTORE *SDATA
160 FOR I=0 TO 16*6-1
170  READ A$:POKE &HBF00+I,VAL("&H"+A$)
180 NEXT
190 *SDATA
200 DATA  01,00,01,ED,5B,F2,BF,2A,F0,BF,F3,D3,5C,ED,B0,00
210 DATA  D3,5F,FB,C9,00,00,00,00,00,00,00,00,00,00,00,00
220 DATA  01,00,01,ED,5B,F2,BF,2A,F0,BF,F3,D3,5D,ED,B0,00
230 DATA  D3,5F,FB,C9,00,00,00,00,00,00,00,00,00,00,00,00
240 DATA  01,00,01,ED,5B,F2,BF,2A,F0,BF,F3,D3,5E,ED,B0,00
250 DATA  D3,5F,FB,C9,00,00,00,00,00,00,00,00,00,00,00,00
260 ´========= Input V-RAM Address =========
270 ADDR=&HC000 : ENDAD=&HFFFF
280 ´========= Get Visual Data to Disk =========
290 ´---- Data Destination Set ----
300 OPEN F$ FOR OUTPUT AS #2 :PAGE=1
310 PUT #2,3*((ENDAD-ADDR)¥256+1)
320 ´----- Destination Set ----
330  BUFFER=VARPTR(#2)+9
340  POKE &HBFF2,VAL("&H"+RIGHT$(HEX$(BUFFER),2))
350  POKE &HBFF3,VAL("&H"+LEFT$(HEX$(BUFFER),2))
360 ´----- V-RAM Top Set -----
370  POKE &HBFF0,VAL("&H"+RIGHT$(HEX$(ADDR),2))
380  POKE &HBFF1,VAL("&H"+LEFT$(HEX$(ADDR),2))
390 ´----- Loop ----
400 FOR I=ADDR TO ENDAD STEP 256
410  A=USR0(8) : PUT #2,PAGE : PAGE=PAGE+1
420  A=USR1(8) : PUT #2,PAGE : PAGE=PAGE+1
430  A=USR2(8) : PUT #2,PAGE : PAGE=PAGE+1
440  POKE &HBFF1,(PEEK(&HBFF1)+1) MOD 256
450 NEXT
460 CLOSE #2
470 END
```

このプログラムで書き出されたファイルをPC-9801でグラフィックVRAMにロードする1例として，フィールド文のバッファの内容をPOKEしていく方法があります。

次にそのプログラムを示します。

PC-9801用 G-VRAMロード

```
1 ´save "G8898
1000 INPUT"FILE NAME ";A$
1010 OPEN A$ AS #1 : SCREEN 0,1
1020 FIELD #1,255 AS BUF$,1 AS B$ : ADR=0
```

```
1030 FOR I=1 TO 63
1040  GET #1,I*3-2 : GOSUB *BLUE
1050  GET #1,I*3-1 : GOSUB *RED
1060  GET #1,I*3   : GOSUB *GREEN
1070  ADR=ADR+256
1080 NEXT
1090 CLOSE : SCREEN 0,0 : END
1100 *BLUE
1110  DEF SEG=&HA800
1120  FOR J=1 TO 255
1130   A=ASC(MID$(BUF$,J,1)) : POKE ADR+J-1,A
1140  NEXT
1150  A=ASC(B$) : POKE ADR+J-1,A
1160  RETURN
1170 *RED
1180  DEF SEG=&HB000
1190  FOR J=1 TO 255
1200   A=ASC(MID$(BUF$,J,1)) : POKE ADR+J-1,A
1210  NEXT
1220  A=ASC(B$) : POKE ADR+J-1,A
1230  RETURN
1240 *GREEN
1250  DEF SEG=&HB800
1260  FOR J=1 TO 255
1270   A=ASC(MID$(BUF$,J,1)) : POKE ADR+J-1,A
1280  NEXT
1290  A=ASC(B$) : POKE ADR+J-1,A
1300  RETURN
```

これでは，いくらPC-9801の高速なBASICでも遅さが気になります。それに青，赤，緑の画面にそれぞれ書き込んでいますので3画面とも書き終わるまでに正常な色がでないことになります。

そこで，ファイルバッファを利用したちょっとしたテクニックとマシン語ルーチンを使って，高速書き込みを行ってみましょう。

BASICの部分は次のとおりです。

高速グラフィックローダー

```
1 'save "HSG"
1000 SCREEN 0,0 :CONSOLE 0,25,0,1
1010 INPUT"FILE NAME ";F$
1020 DEF SEG=&H1F00
1030 FOR I=0 TO &H78
1040  READ D$:D=VAL("&H"+D$)
1050  POKE I,D
1060 NEXT I
1070 A=0 : B=&H3F :AD%=0
1080 OPEN F$ AS #1
1090 OPEN F$ AS #2
1100 OPEN F$ AS #3
1110 O1%=VARPTR(#1,0)+&H20 : S1%=VARPTR(#1,1)
1120 O2%=VARPTR(#2,0)+&H20 : S2%=VARPTR(#2,1)
1130 O3%=VARPTR(#3,0)+&H20 : S3%=VARPTR(#3,1)
1140 CALL A(S1%,O1%,S2%,O2%,S3%,O3%)' First Call
```

```
1150 FOR I=1 TO 189 STEP 3
1160  GET #1,I
1170  GET #2,I+1
1180  GET #3,I+2
1190  CALL B(AD%)  ' Second Call
1200  AD%=AD%+256
1210 NEXT I
1220 CLOSE :END
1230 DATA BF,33,00,C4,37,E8,1F,00,C4,77,04,E8,19,00,C4,77
1240 DATA 08,E8,13,00,C4,77,0C,E8,0D,00,C4,77,10,E8,07,00
1250 DATA C4,77,14,E8,01,00,CF,26,8B,04,1E,0E,1F,89,05,47
1260 DATA 47,1F,C3,A8,22,60,00,80,22,60,00,58,22,60,00,C4
1270 DATA 37,26,8B,14,0E,1F,BB,33,00,1E,C5,37,8B,34,B8,00
1280 DATA B8,E8,1A,00,1F,1E,C5,77,04,8B,34,B8,00,B0,E8,0D
1290 DATA 00,1F,C5,77,08,8B,34,B8,00,A8,E8,01,00,CF,8E,C0
1300 DATA B9,80,00,89,D7,FC,F3,A5,C3,00,00,00,00,00,00,00
```

1つのファイルを3つのファイル番号でオープンし，それぞれのファイルバッファのアドレスを求めています。そのアドレスをマシン語ルーチンに引き渡した後，GET してファイルバッファにデータを読み込んでいます。その後，マシン語ルーチンで，ファイルバッファからグラフィック VRAM に転送するという方法です。マシン語ルーチンは次のとおりです。

```
                        ;*********************************************
                        ;* HIGH SPEED LOADER
                        ;* PC-8801->PC-9801 GRAPHICS DATA
                        ;*********************************************
                        ;
                        ;=============================================
                        ; FIRST CALL
                        ;     Calling sequence
                        ;        CALL A(S1%,O1%,S2%,O2%,S3%,O3%)
                        ;     Return code
                        ;        NONE
                        ;=============================================
                        ;
0000 BF3300             GETP:  MOV DI,OFFSET DATA
0003 C437                      LES SI,[BX]
0005 E81F00      0027          CALL LOAD         ;6th param. O3%

                        ;
0008 C47704                    LES SI,04H[BX]
000B E81900      0027          CALL LOAD         ;5th param. S3%

                        ;
000E C47708                    LES SI,08H[BX]
0011 E81300      0027          CALL LOAD         ;4th param. O2%

                        ;
0014 C4770C                    LES SI,0CH[BX]
0017 E80D00      0027          CALL LOAD         ;3rd param. S2%

                        ;
001A C47710                    LES SI,10H[BX]
001D E80700      0027          CALL LOAD         ;2nd param. O1%
```

```
                                ;
0020 C47714                                 LES SI,14H[BX]
0023 E80100          0027                   CALL LOAD         ;1st param. S1%

                                ;
0026 CF                                     IRET              ; BACK TO BASIC
                                ;
                                ;----------------------------------------
0027 268B04                     LOAD:       MOV AX,ES:[SI]
002A 1E                                     PUSH DS           ; SAVE DS
002B 0E                                     PUSH CS
002C 1F                                     POP DS            ; DS=CS
002D 8905                                   MOV [DI],AX       ; STORE param.
002F 47                                     INC DI
0030 47                                     INC DI
0031 1F                                     POP DS            ; RESTORE DS
0032 C3                                     RET               ; RETURN TO MAIN
                                ;----------------------------------------
                                ;
0033                            DATA        RW 06H
                                ;
                                ;========================================
                                ; SECOND CALL
                                ;     Calling sequence
                                ;         CALL B(AD%) : AD%=DEST OFFSET
                                ;     Return code
                                ;         NONE
                                ;========================================
                                ;
003F C437                       SEC:        LES SI,[BX]
0041 268B14                                 MOV DX,ES:[SI]    ; DX=AD%=DESTINATION
0044 0E                                     PUSH CS
0045 1F                                     POP DS            ; DS=CS
0046 BB3300                                 MOV BX,OFFSET DATA
                                ;
0049 1E                                     PUSH DS           ; SAVE DS
004A C537                                   LDS SI,[BX]       ; DS=S3%:SI=03%
004C 8B34                                   MOV SI,[SI]       ; SI=SOURCE OFFSET AD
004E B800B8                                 MOV AX,0B800H     ; AX=GREEN
0051 E81A00          006E                   CALL TRANS        ; TRANSFER
                                ;
0054 1F                                     POP DS            ; RESTORE DS
0055 1E                                     PUSH DS           ; SAVE DS
0056 C57704                                 LDS SI,4[BX]      ; DS=S2%,SI=02%
0059 8B34                                   MOV SI,[SI]       ; SI=SOURCE OFFSET AD
005B B800B0                                 MOV AX,0B000H     ; AX=RED
005E E80D00          006E                   CALL TRANS        ; TRANSFER
                                ;
0061 1F                                     POP DS            ; RESTORE DS
0062 C57708                                 LDS SI,8[BX]      ; DS=S1%,SI=01%
0065 8B34                                   MOV SI,[SI]       ; SI=SOURCE OFFSET AD
0067 B800A8                                 MOV AX,0A800H     ; AX=BLUE
006A E80100          006E                   CALL TRANS        ; TRANSFER
                                ;
006D CF                                     IRET              ; BACK TO BASIC
                                ;
                                ;----------------------------------------
                                ;
006E 8EC0                       TRANS:      MOV ES,AX         ; ES = GVRAM SEGMENT
0070 B98000                                 MOV CX,128        ; CX = WORD TRANSFER
```

```
0073 8BFA                 MOV DI,DX       ; DI = DESTINATION
0075 FC                   CLD             ; INCREMENT
0076 F3A5                 REP MOVSW       ; MOVE WORD
0078 C3                   RET             ; RETURN TO MAIN
                     ;-----------------------------------------
```

第 12 章　RS-232C

12-1　RS-232Cとは
12-2　専用ケーブルの作り方
12-3　通信モードの指定
12-4　プログラムの転送
12-4-1　メモリー上にある場合
12-4-2　ディスクファイルにある場合
12-5　コミュニケーション・プログラム

第12章　RS-232C

12-1　RS-232Cとは

PC-9801は，RS-232Cインターフェイスを内蔵しておりシリアルデータの送受信が行えます。RS-232Cとは米国のEIA（Electronic Industries Association）で規定されたシリアルデータのインターフェイスで，日本でもJIS C6361-71として制定されています。

PC-9801ではこのインターフェイスを使って，RS-232Cインターフェイス付きの機器とのデータ交換ができます。データ交換は，ターミナルモードと入出力モードの2通りの方法があります。ターミナルモードでは大型計算センターのTSS端末やオンラインシステムの端末として利用でき，入出力モードでは，制御機器，計測機器，他のパーソナルコンピュータなどとデータの交換ができます。

この章では，入出力モードによるデータ交換をとりあげパーソナルコンピュータ同士でプログラムやデータの送信・受信を行うことについて考えてみたいと思います。ここでは，実際にPC-9801とPC-8801とを接続してデータ交換を行ってみることにします。

12-2　専用ケーブルの作り方

PC-9801と他の機種とを接続するためには専用ケーブルが必要です。専用ケーブルはRS-232C用オスコネクタ（D SUB 25ピン・オスコネクタ）が2個と10芯程度のマイクコードと呼ばれる多芯シールドケーブルがあれば作ることができます。

RS-232C用のコネクタには，25個のピンがありますが，実際はこれらをすべて使うわけではありません。PC同士の接続だけなら極端な場合，3本の信号線があればデータ交換ができます。その3本の信号線は，送信データ，受信データ，信号接地の各線です。ここで注意することは，送り手の送信データは受け手の受信データに接続しなければデータ交換ができないことです。というのは，送信データと受信データの信号線は送り手と受け手では逆になっているということです。

これはRS-232Cという規格がコンピュータとモデムやカプラを介して他のコンピュータや機器に接続するためのものであるからです。そのため，送信データ・受信データという信号線は，コンピュータからモデムやカプラを見ているかぎり不自然ではありません。しかし，直接コンピュータ同士を接続するときには，一方の送信は他方の受信というようにピンを入れ替える必要があるわけです。

次に，専用ケーブルの接続の方法を図12-2に示します。

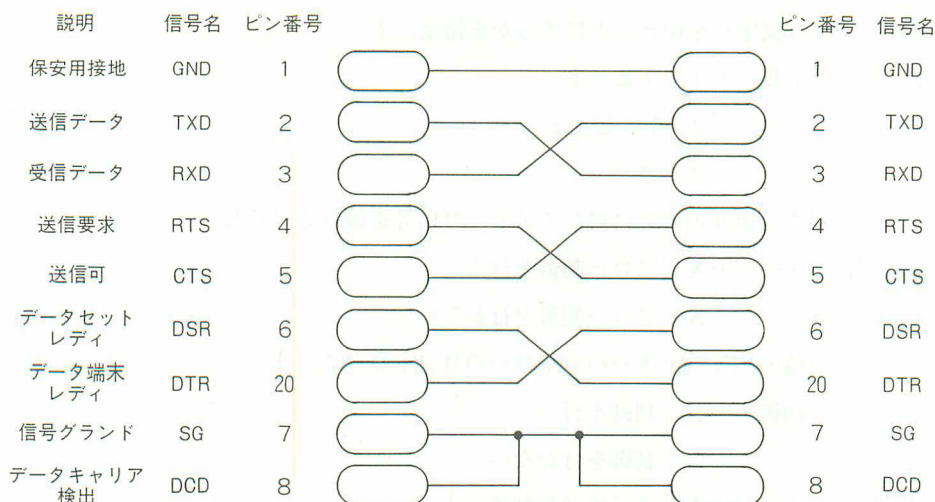



図 12-2 専用ケーブル接続図

これでPC同士が接続できます。対になる信号を入れ替えたことで一方から他方がモデムの様に見えるためです。

12-3 通信モードの指定

N_{88}-BASIC (86) の入出力処理は，ファイルの処理と同じ考え方に統一されています。RS-232 C インターフェイスで扱うデータもファイルという概念で取り扱われています。そのため，RS-232 C インターフェイスファイルは，ファイルディスクリプタを持っています。デバイス名は〝COM：〃で，それに続いて通信時のデータ形式や制御情報を指定します。

OPEN 〝COM：①②③④⑤〃 …

通信形式は5文字のパラメータによって設定します。次にパラメータの意味と指定できる文字を一覧表として示します。

① パリティ………E(ven)：偶数パリティチェック
　　　　　　　　O(dd)：奇数パリティチェック
　　　　　　　　N(one)：パリティチェックなし

（binary codeの〝1〃の数を奇数もしくは偶数にするように余分のbitを付加し，そのbinary codeの誤りをチェックする。）

② データビット長………7：7ビット
　　　　　　　　　　　　8：8ビット

〔 1ワード（1文字）を何ビットにするかを指定。〕

③ ストップビット長……1：1ビット

2：1.5ビット

3：2ビット

〔 字信号に対する個々の符号に対してストップ信号を後行させるビット。〕

④ フロー制御……………X：フロー制御を行う

N：フロー制御を行わない

〔 データ受信時のバッファオーバーフローのコントロール。〕

⑤ シフトコード制御……S：制御を行う

N：制御を行わない

〔 英数字やカナコードなどを示す通信制御。〕

注1．④のフロー制御を指定した場合，データ受信時にバッファ（256文字）の残りが23文字分になると，システムはCTRL-Sのコード（19）を出して相手側に通信の一時停止を要求し，その後バッファが空になった時点でCTRL-Qのコード(17)を出力して送信再開を許可します。

注2．データビット長が7ビットの場合，カナ文字を送受するためには⑤のシフトコード制御を行う必要があります。

例えば，

```
OPEN "COM:E81XN" FOR OUTPUT AS #1
```

とすると，偶数パリティ，データビット長が8ビット，ストップビット長が1ビット，フロー制御を行い，シフトコード制御は行わないという指定になり，送信する準備ができたことになります。

もう1つ大事な指定があります。それは通信速度(ボーレート)です。これは，メモリスイッチの2(SW 2)で行い次のようになります。

通信速度…………1：　75ボー

2：　150ボー

3：　300ボー

4：　600ボー

5：1200ボー

6：2400ボー

7：4800ボー

8：9600ボー

〔 1秒間に転送するビット数。〕

PC-9801ではシステム既定値として1200ボーが設定されています。ボーレートの設定はモニタモードで次のようにします。例として600ボーにセットしてみます。

```
MON
h]SSW ← スイッチの状態を調べる
SW1 SW2 SW3 SW4 SW5 SW6 SW7
48  05  00  00  00  00  00
h]SSW2 ← SW2の内容を4にする
05-04
h]SSW ← セットされたか確認
SW1 SW2 SW3 SW4 SW5 SW6 SW7
48  04  00  00  00  00  00
h]^B
Ok
```

なお，PC-9801本体背面のディップスイッチの2の5番目をONにしておくとメモリスイッチの内容は電源をOFFにしても保存されます。

12-4 プログラムの転送

PC-8801からPC-9801にBASICのプログラムを送ることにし，通信形式は次のように指定するものとします。

通　信　速　度：600ボー（4）
パ　リ　テ　ィ：偶数パリティ（E）
データビット長：8ビット（8）
ストップビット長：1ビット（1）
フ　ロ　ー　制　御：行う（X）
シフトコード制御：行う（S）

PC-8801ではボーレートの設定は，本体背面のジャンパースイッチで行います。600ボーにするにはその4番目にセットします。

12-4-1 メモリー上にある場合

プログラムがメモリー上にあるときには，SAVE・LOADだけで送受信が可能です。

① PC-9801で，

　LOAD "COM：E81XS" [RET]

　とします。

SAVE "COM：E81XS"[RET]
OPEN "COM：E81XS" FOR OUTPUT AS #1：
PRINT #1, CHR$(4)：CLOSE

とします。

PC-8801でSAVE "COM：E81XS" とすると，転送フォーマットはアスキー形式となります。これはディスクにアスキーセーブするのと同じ形式です。そのため，PC同士のBASICの中間言語が異なっていても受信にはまったく差しつかえありません。1つディスクへのアスキーセーブと違うのは，送信した際，プログラムの終りを示すエンドマークがないことです。

一方，PC-9801側では，LOADのときにRS-232Cから送られてくる入力に対して，人間がキー入力するのと全く同じ動作でプログラムを格納していきます。そしてプログラムの最後の行を受信しても，送信側でエンドマークを送っていませんので，いつまでたってもLOADコマンドから抜け出ません。そこで，PC-8801側で，ファイルをオープンして転送終了のコード(ET)，CHR$(4)を送っています。この信号を受けるとPC-9801は，"Direct statement in file" のエラーが出ますが，正常にプログラムを受信して，コマンド待ちになります。

12-4-2 ディスクファイルにある場合

こんどはディスクからディスクへの転送を行ってみましょう。

N_{88}-BASICのプログラムを転送する前に，それがアスキーセーブされているかを確認して下さい。次のプログラム①をPC-9801に入れ，プログラム②をPC-8801に入れて，PC-9801から先に実行します。すると，PC-9801のディスクにPC-8801のプログラムがアスキーセーブされた形になります。

プログラム① (PC-9801)

```
1 'save "FROM88.bas"
100 ' Receive from  PC-8801
110 INPUT "Fle name ";F$
120 OPEN F$ FOR OUTPUT AS #1
130 OPEN "COM:E81XS" FOR INPUT AS #2
140  LINE INPUT #2,A$
150  IF A$=CHR$(4) THEN *DEND
160 PRINT A$
170 PRINT #1,A$
180 GOTO 140
190 '
200 *DEND
210  CLOSE #1,2
220 END
```

プログラム② (PC-8801)

```
1 'save "T098.bas"
100 ' Transfer to PC-9801
110 INPUT "Fle name ";F$
120 OPEN F$ FOR INPUT AS #1
130 OPEN "COM:E81XS" FOR OUTPUT AS #2
140  IF EOF(1) THEN *DEND
150  LINE INPUT #1,A$
160 PRINT A$
170 PRINT #2,A$
180 GOTO 140
190 '
200 *DEND
210 PRINT #2,CHR$(4)
220  CLOSE #1,2
230 END
```

なお，データファイルの転送も同じプログラムで送受信ができます。

12-5 コミュニケーション・プログラム

この章のまとめとして，PC-9801上で動くファイル転送プログラムを紹介します。PC-9801同士やPC-9801とホストコンピュータを接続してRUNすれば，お互いにデータ交換が可能です。

```
1 'save "TSS.bas" :'TSS TERMINAL
100 SCREEN 0,3:WIDTH 80,25:CLEAR ,&H1D00:DEFINT A-Z
110 DEF SEG = &H1D00:FOR I=0 TO &H33
115  READ DA$:POKE I,VAL("&h"+DA$):NEXT
120 CSR.LOCATE = 0 : CSR.DISP = &H2D : GOTO 170
130 DATA 50,52,06,56,8B,77,06,8E,C6,8B,77,04,26,8A,14,8B
140 DATA 77,02,8E,C6,8B,37,26,8A,34,B8,A0,00,F6,E6,32,F6
150 DATA 02,D2,03,D0,B4,13,CD,18,5E,07,5A,58,CF,50,B4,11
160 DATA CD,18,58,CF
170 OPEN "COM1:E81" AS 1:WD=80:CONSOLE ,,0,0
180 FOR I=1 TO 10:KEY I,"":NEXT
190 KEY 1,"D-load" :KEY 2,"U-load"
195 KEY 3,"40-80" :KEY 4,"Exit" :KEY ON
200 *MAIN
210 WIDTH WD,25:CONSOLE 3,25,1,1:LOCATE 0,0
220 PRINT "----------------------------------------"
230 PRINT " Communications Program  Terminal Mode"
240 PRINT "----------------------------------------"
250 '
260 ON KEY GOSUB *PC.TO.HOST,*HOST.TO.PC,*WIDTH.CHANGE,*EXIT
270 ON STOP GOSUB *STOP.KEY.IN:STOP ON
280 '--- cominucatin program by RS-232C ---
290 WHILE 1
300 *MAIN2
310   X=POS(0):Y=CSRLIN:CALL CSR.LOCATE(X,Y)
315   CALL CSR.DISP:C$=INKEY$
320   WHILE C$<>"" : PRINT #1,C$; : C$="" : WEND
330   WHILE LOC(1)<>0 : PRINT INPUT$(LOC(1),#1); : WEND
```

```
340 WEND
350 '
360 *STOP.KEY.IN:STOP OFF:C$=CHR$(3):STOP ON:RETURN 320
370 '
380 '--------------------------------------------
390 '     File Transmit Prgram to HOST from PC
400 '--------------------------------------------
410 *PC.TO.HOST:KEY OFF     : CR$=CHR$(13):LF$=CHR$(10)
420     ON ERROR GOTO 510
430     ON STOP GOSUB *RET:STOP ON
440     LOCATE 26,1:PRINT "File Out Mode ":LOCATE 0,24:PRINT
450     INPUT "file name";F$:OPEN F$ FOR INPUT AS #2
460         C$=INPUT$(1,#2):IF EOF(2) THEN 490 ELSE PRINT C$;
             :PRINT #1,C$;
470         IF C$=CR$ THEN PRINT #1,LF$
480         GOTO 460
490     GOTO *RET
500
510 IF ERR=53 THEN RESUME 530
520 GOTO  450
530 PRINT:PRINT F$;" Not Found.":BEEP:PRINT:FILES:GOTO 450
540 '
550 '--------------------------------------------
560 '     File Transmit Program to PC from HOST
570 '--------------------------------------------
580 *HOST.TO.PC:KEY OFF
590 LOCATE 26,1:PRINT "File in  Mode ":LOCATE 0,24:PRINT
600 ON STOP GOSUB *RET:STOP ON
605 INPUT "file name ";F$:OPEN F$ FOR OUTPUT AS 2
610 X=POS(0):Y=CSRLIN:CALL CSR.LOCATE(X,Y)
615 CALL CSR.DISP:C$=INKEY$
620 IF C$=""THEN 630 ELSE PRINT #1,C$;:C$=""
630 IF LOC(1)<>0 THEN S$=INPUT$(LOC(1),#1):
      PRINT S$;:PRINT #2,S$;
640 GOTO 610
650 '
660 *RET:RETURN *RET2
670 *RET2:CLOSE 2:STOP OFF:KEY ON:CLS:LOCATE 26,1
675  PRINT "Terminal Mode ":LOCATE 0,3:RETURN *MAIN
680 '
690 *WIDTH.CHANGE:KEY OFF
700     IF WD=40 THEN WD=80 ELSE WD=40
710     GOTO *RET2
720 '
730 *EXIT
740 CONSOLE ,,0,0
750 KEY 1,"load "+CHR$(&H22) :KEY 2,"auto "
760 KEY 3,"go to "           :KEY 4,"list "
770 KEY 5,"run"+CHR$(13)     :KEY 6,"save "+CHR$(&H22)
780 KEY 7,"key "             :KEY 8,"print "
790 KEY 9,"edit ."           :KEY 10,"cont"+CHR$(13)
800 CONSOLE 0,24,1,1:WIDTH 80
810 KEY OFF
820 PRINT "Communications End! Good-by!!":END
```

第 13 章　PC-9801F

13-1　システム概要

13-2　5インチ倍トラックディスク

13-3　漢字ROMと日本語BASIC

13-4　拡張グラフィック画面

13-5　拡張ステートメント

13-6　PC－9801E

第13章　PC-9801F

本書では各章を通じて，PC-9801とその姉妹機PC-9801 Fの両方のマシンに対して共通に解説し，プログラム等を紹介しています。しかし，PC-9801 Fは，PC-9801と異なる機能と特徴を持っているためすべてを包括して説明することはできません。そこでこの章では，PC-9801 Fに関して特にPC-9801との相異点をクローズアップしています。

13-1　システム概要

PC-9801 FはPC-9801に比べて次の点が大きく異なっています。

① クロック8 MHzの8086を搭載。
② 5インチ倍トラックのフロッピィディスクを内蔵（PC-9801 F 1……1台，PC-9801 F 2……2台）。
③ 漢字ROM（JIS第1水準）を標準装備。
④ N_{88}-日本語BASIC (86) を標準装備。
⑤ グラフィックVRAMを拡張（カラーの640×400ドットが2画面）。
⑥ N_{88}- BASIC (86) がバージョンアップされて2.0となっています。ステートメントがいくつか追加・拡張されています。

13-2　5インチ倍トラックディスク

PC-9801 Fは，本体に5インチ両面倍密度・倍トラックのフロッピィディスクを内蔵しています。1台内蔵されているのがPC-9801 F 1で，2台内蔵されているのがPC-9801 F 2と呼ばれます。

容量は1台で従来の5インチの2倍，つまり640 Kバイトあります。これは，トラック数がいままでの40トラックから80トラックに増えているためです。

また，転送方式が8インチと同じDMA方式になったためPC-8031-2 WやPC-80 S 31-2 Wのインテリジェント型に比べ，アクセスのスピードが大幅にアップされています。

なお，PC-9801の5インチ・ディスケットに収められているプログラムやデータは，付属のユーティリティプログラムDDconv.n 88で，倍トラック用に変換することができます。

13-3　漢字ROMと日本語BASIC

漢字ROMはPC-9801ではオプションですがPC-9801 Fでは標準装備になっています。これに伴ない，ディスクシステムでは，N_{88}-日本語BASIC (86) が付加されています。そのシステムディスクには，mkfont.n 88とusfont.n 88というユーティリティプログラムとフォントデータファイ

ルがあり，ユーザーが7621H～765FHの漢字コードに対応する部分を自由に定義して，ファイルに登録・更新ができます。usfont.n 88 のフォントデータはシステム起動時に自動的にロードされます。なお，日本語BASICを起動したくないときは，メモリスイッチ6を1にしておきます。

13-4 拡張グラフィック画面

PC-9801 Fは，グラフィックVRAMがPC-9801に比べ2倍に拡張されています。つまり，96 Kバイトの G-VRAMが2組あります。このため，640×400 ドットのカラーモードで2画面使えます。これらのG-VRAMはメモリマップ上A 8000 Hから図13-4のように同じアドレスに割り当てられており，バンク切り換えで表示・アクセスする画面を選択するようになっています。

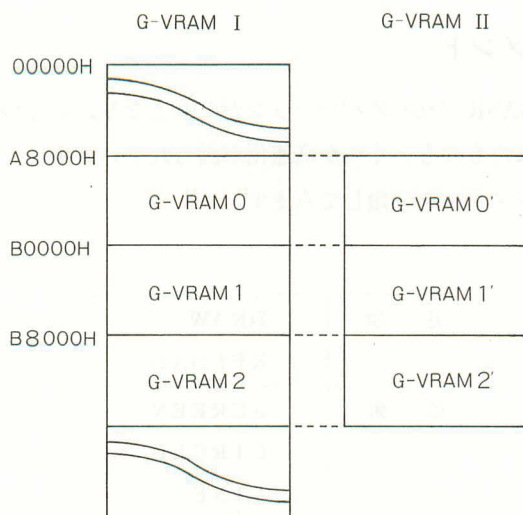



図 13-4 G-VRAM のメモリマップ

この2組のG-VRAMのどちらをアクセス可能にするかは，I/Oポートの A 6 H を，またどちらの G-VRAM を表示するかは，A 4 H を用いて制御されます。

次に，アセンブリ言語でのセレクト方法を示します。なお，これは，BASIC の OUT 文でも実行できます。 コメントを参照して下さい。

G-VRAM I

● アクセス

```
MOV     AL,0
OUT     A6H,AL      ;OUT &HA6,0
```

● 表　示

```
MOV   AL,0
OUT   A4H,AL   ;OUT &HA4,0
```

G-VRAM II

● アクセス

```
MOV   AL,1
OUT   A6H,AL   ;OUT &HA6,1
```

● 表　示

```
MOV   AL,1
OUT   A4H,AL   ;OUT &HA4,1
```

13-5 拡張ステートメント

PC-9801Fでは，N_{88}-BASIC (86) のバージョンが2.0となり，いくつかのステートメントが追加，拡張され，さらに従来のものもいくつか高速化が画られています。

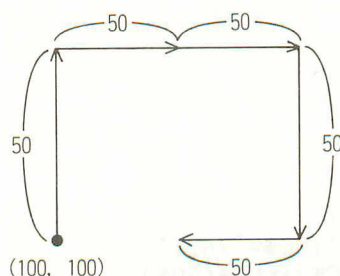
次に，これらのステートメントを分類してみます。

追　加	DRAW KPLOAD
拡　張	SCREEN CIRCLE LINE ROLL
高速化	CLS 2, 3

これらの使い方を簡単に説明します。

● DRAW………グラフィック画面サブコマンドによる連続的な図形描画機能。

直線を組み合わせた図形が簡単に描けます。これはLOGOという言語のタートルグラフィックスの描き方と共通する点があります。

ＰＯＩＮＴ（１００，１００）
例）DRAW "U50 R50 R50 D50 L50"

(100, 100) から Up, Right, Right, Down, Left へそれぞれ50ドットずつ線を描きます。もちろん，これらの指示を文字列に代入してもOKです。

A$= "U50 R50 R50 D50 L50"

ＤＲＡＷ　Ａ＄

● KPLOAD………ユーザー定義フォントパターンの登録機能。

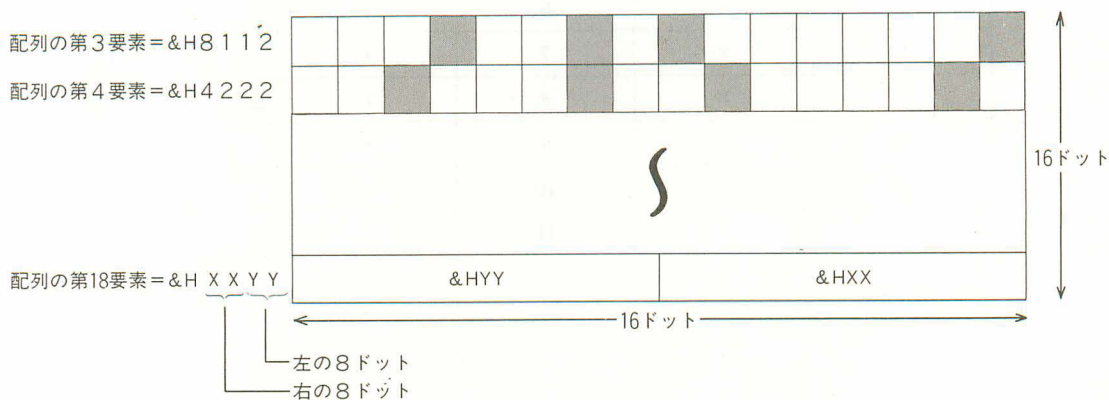
例）KPLOAD &H 7621, FP%

漢字コード　フォントパターン

漢字コードのフォントパターンとして，整数型配列名で指定されたものを登録します。漢字コードは7621 H〜765 FHまで使用可能です。配列には次のように指定します。なお，これは

配列の第1要素＝16

配列の第2要素＝16

配列の第3要素から第18要素まで次のようにフォントパターンのドットイメージを格納します。なおこれはmkfont.n88のプログラムでドットイメージを作れば，その16進コードが簡単に得られます。

次に簡単なサンプルプログラムを示します。

```
1 ' save "kpload"
100  '-- KPLOAD Sample --
110 DIM FP%(17)
120 FP%(0)=16:FP%(1)=16
130 FOR I=2 TO 17
140   READ FP$
150   FP=VAL("&H"+FP$)
160   FP%(I)=FP
170 NEXT I
180 'KPLOAD &H7621,FP%
190 SI$="1B4B":SO$="1B48":FC$="7621"
200 KJ$=KNJ$(SI$)+KNJ$(FC$)+KNJ$(SO$)
210 PRINT KJ$
220 END
230 ' Font Pattern Data
240 DATA FF7F,FF7F,FF7F,781E,781E,781E,781E,781E
250 DATA 781E,781E,781E,781E,781E,FF7F,FF7F,FF7F
```

● SCREEN

G-VRAMが2倍に拡張されたためアクティブページとディスプレイページの指定が2倍に増えています。次にその指定一覧表を示します。

ページ番号	画面モードごとのアクティブページ指定値			
	0	1	2	3
1	0	0	0	0
2	1	1	1	1
3	2	2	2	×
4	3	3	3	×
5	×	4	4	×
6	×	5	5	×
7	×	6	×	×
8	×	7	×	×
9	×	8	×	×
10	×	9	×	×
11	×	10	×	×
12	×	11	×	×

×印：指定不可

ディスプレイページの値	画面モードごとの意味			
	0	1	2	3
0	全ページ表示しない	全ページ表示しない	全ページ表示しない	全ページ表示しない
1	ページ１のみ表示	ページ１のみ表示	ページ１のみ表示	ページ１のみ表示
2	ページ２のみ表示	ページ２のみ表示	ページ２のみ表示	×
3	×	ページ１，２を合成表示	ページ１，２を合成表示	×
4	×	ページ３のみ表示	ページ３のみ表示	×
5	×	ページ１，３を合成表示	ページ１，３を合成表示	×
6	×	ページ２，３を合成表示	ページ２，３を合成表示	×
7	×	ページ1,2,3を合成表示	ページ1,2,3を合成表示	×
8	全ページ表示しない	全ページ表示しない	全ページ表示しない	全ページ表示しない
9	×	ページ４のみ表示	×	×
10	×	ページ５のみ表示	×	×
11	×	ページ４，５を合成表示	×	×
12	×	ページ６のみ表示	×	×
13	×	ページ４，６を合成表示	×	×
14	×	ページ５，６を合成表示	×	×
15	×	ページ4,5,6を合成表示	×	×
16	全ページ表示しない	全ページ表示しない	全ページ表示しない	全ページ表示しない
17	ページ３のみ表示	ページ７のみ表示	ページ４のみ表示	ページ２のみ表示
18	ページ４のみ表示	ページ８のみ表示	ページ５のみ表示	×
19	×	ページ７，８を合成表示	ページ４，５を合成表示	×
20	×	ページ９のみ表示	ページ６のみ表示	×
21	×	ページ７，９を合成表示	ページ４，６を合成表示	×
22	×	ページ８，９を合成表示	ページ５，６を合成表示	×
23	×	ページ7,8,9を合成表示	ページ4,5,6を合成表示	×
24	全ページ表示しない	全ページ表示しない	全ページ表示しない	全ページ表示しない
25	×	ページ10のみ表示	×	×
26	×	ページ11のみ表示	×	×
27	×	ページ10,11を合成表示	×	×
28	×	ページ12のみ表示	×	×
29	×	ページ10,12を合成表示	×	×
30	×	ページ11,12を合成表示	×	×
31	×	ページ10,11,12を合成表示	×	×

● CIRCLE………円を描くと同時にその内部を塗りつぶす機能が拡張されています。

書式は次のように最後にFを付けてペイント指定を行います。

```
CIRCLE | (Wx, Wy)      | ,〈半径〉, [,パレット1]
       | STEP (X, Y)   |

[,〈開始角度〉] [,〈終了角度〉] [,〈比率〉] [,F [, | 〈パレット2〉       | ]]
                                              | 〈タイルストリング〉 |
```

例）CIRCLE (80,80),50,4,0,5,1,F,1

● LINE……BOX 指定時に塗りつぶしがタイルパターンで行えます。

次に書式を示します。

```
LINE [ | (Wx1, Wy1)      | ] - | (Wx2, Wy2)      |
       | STEP (X1, Y1)   |     | STEP (X2, X2)   |

[,〈パレット1〉] [, | B  | ] [, | 〈ラインスタイル〉   | ]
                    | BF |     | 〈パレット2〉       |
                               | 〈タイルストリング〉 |
```

例）T$=CHR$(&H12)+CHR$(&H55)+CHR$(0)

LINE (0,0)−(100,100),7,BF,T$

● ROLL（上下左右方向のロール機能が追加）

上下方向は±399ドット，左右方向は±639ドットの範囲でスクロールできます。また，スクロールした後，新たに表われた領域を0でクリアするか（Nを指定）バックグラウンドカラーでクリアするか（Yを指定）を指定することができます。

次に書式を示します。

```
ROLL [〈上下〉 [,〈左右〉] [, | Y | ]]
        ↑           ↑          | N |
   方向ドット数  方向ドット数
```

例）ROLL 50,−50,Y

● CLS 2，CLS 3

オールクリアのスピードがPC-9801に比べ，約13〜40倍も速くなっています。

13-6 PC-9801E

PC-9801のもう1台の姉妹機にPC-9801Eというのがあります。これは，PC-9801Fとハードウェアのアーキテクチャが同じで，ソフトウェアもコンパチブルになっています。大きな違いは，2DDのディスクと漢字ROMが内蔵されておらず，デザイン的にはPC-9801と似ており，コンパクトになっていることです。そしてディスクのインターフェイスは，5インチ2D用しか付いていません。8インチのディスクドライブを接続するには別売のインターフェイスボードが必要です。5インチ2DDも接続可能ですが，これもインターフェイスボードが必要です。また，N_{88}-BASIC(86)のROMはマザーボードにあるため，拡張スロットは6個フルに使用できます。ちなみに，PC-9801Eの拡張スロット6とPC-9801Fの拡張スロット4は，8インチディスクのインターフェイスボード専用になっていますので，ボード取り付けの際には注意して下さい。

なお，本書で紹介していますプログラムはPC-9801Eでも作動します。

第 14 章　ランダムテクニック

14-1　行番号0

14-2　2バイトの数字を上位・下位の1バイトに分ける

14-3　REM文の効率

14-4　エラーメッセージをすべて表示するには

14-5　マシン語でエラーメッセージを表示

14-6　未使用コマンドを使用する

14-7　新しいコマンドを作る

14-8　8086はリセットがかかったら何処へ?!

14-9　INKEY$でカーソル表示

14-10　高速リスト

14-11　CHR$(13);CHR$(10)とCHR$(13)+CHR$(10)との違い

14-12　OUTPUTとASも変数に使える

14-13　キーバッファクリア

14-14　リアルタイムで時間表示

14-15　モニタモードでファンクションキーを使用する

第14章　ランダムテクニック

14-1　行番号0

BASICプログラムの行番号は，1～65529までの整数というのが普通ですが，N_{88}-BASIC (86) でも，行番号0が使えます。ただし，スクリーンエディット時には，行番号0は使えませんので，次のように，間接的に行番号0のテキストを作ります。

まず，1以上の行番号を持ったテキストを入力します。次にRENUM 0を実行すれば，行番号0のテキストができるわけです。

この場合，1つの行しか行番号を0にできませんが，次の方法では，複数行の行番号を0にすることができます。

これはRAM上にあるプログラムテキストの行番号を，直接0にしてしまうもので，一つ間違うとプログラム自体をこわしてしまったり，リセットしてしまう可能性がありますので，注意が必要です。

具体的な例をみてみましょう。

```
10 PRINT "This line number is 0."
20 PRINT "This is also line 0."
30 PRINT "Oh ! Here is also zero !"
```

モニタモードで，メモリ内容をダンプし，行番号の部分を0にします。

```
MON
hJC60
hJD6A4,6A7
06A4 B0 26 11 27                                          -& ´
hJD26B0,2711
26B0 20 00 0A 00 01 C0 01 22 54 68 69 73 20 6C 69 6E     タ "This lin
26C0 65 20 6E 75 6D 62 65 72 20 69 73 20 30 2E 22 00 e number is 0."
26D0 1E 00 14 00 01 C0 01 22 54 68 69 73 20 69 73 20     タ "This is
26E0 61 6C 73 6F 20 6C 69 6E 65 20 30 2E 22 00 23 00 also line 0." #
26F0 1E 00 01 C0 01 22 4F 68 20 21 20 48 65 72 65 20     タ "Oh ! Here
2700 69 73 20 61 6C 73 6F 20 7A 65 72 6F 2E 20 21 22 is also zero. !"
2710 00 00
hJ
hJS26B2
26B2 0A-00
hJS26D2
26D2 14-00
hJS26F0
26F0 1E-00
hJ^B
Ok
```

これで完成です。もちろん実行もできます。

```
0 PRINT "This line number is 0."
0 PRINT "This is also line 0."
0 PRINT "Oh ! Here is also zero !"
run
This line number is 0.
This is also line 0.
Oh ! Here is also zero !
Ok
```

さて，上限の65529は16進表現でFFF9Hですから，まだ上があります。では，行番号をFFFAH～FFFFH（65530～65535）の方を手で書き直して作ったらどうなるかというと，FFFFH以外はリストをしても表示され，65530～65534の行番号を作ることができ，実行もできます。ただし，GOTOやGOSUBなどで，行番号を参照することはできません。

```
Undefined line number
```

のエラーが出ます。ただし，ラベルでの参照はできます。

特に，FFFFH（65535）にした行がどうしてリストでは表示されないかというと，BASICインタプリタではFFFFHをダイレクトモードのフラグとして扱っているからなのです。

ただし，このようにしてできた行番号0のあるプログラムをアスキーSAVE，

```
SAVE "<ファイル名>", A
```

しますと，LOADする時，

```
Syntax error
```

となってLOADできなくなりますので注意して下さい（MERGEも同じ）。ただし，この場合，

```
OPEN "<ファイル名>" FOR INPUT AS #1
LINE INPUT #1, A$ : PRINT A$
```

とシーケンシャルオープンすれば内容がとり出せます。

14-2　2バイトの数字を上位・下位の1バイトに分ける

POKE文では，1バイトしかメモリに書くことができないので，2バイトの数字，例えば，1234 Hなどは，上位・下位の1バイトずつに分けてPOKEしなければなりません。次にその方法を示します。

《方法1》　(ただし，0000～7FFFH)

```
¥(整数除算),mod(余り) を使う。
A=&H1234
Ok
? HEX$(A ¥ 256)  :'上位1バイト
 12
Ok
? HEX$(A mod 256):'下位1バイト
 34
Ok
```

《方法2》　(0000～FFFFH)

```
INT 宣言して，メモリから直接読む。
DEFINT A
Ok
A=&H1234
DEF SEG=VARPTR(A,1)
Ok
? HEX$(PEEK(VARPTR(A)));'下位1バイト
 34
Ok
? HEX$(PEEK(VARPTR(A)+1)) ;'上位1バイト
 12
Ok
```

《方法3》　(0000～FFFFH)

4桁の文字列にして，分解する。

```
A=&H1234
Ok
A$=RIGHT$("000"+HEX$(A),4)
Ok
? MID$(A$,1,2) ;'上位2ケタ
 12
Ok
? MID$(A$,3,2) ;'下位2ケタ
 34
Ok
```

MID$で抽出した場合は文字列ですから,

```
POKE &H0000,VAL("&H"+MID$(A$,3,2))
```

POKEするアドレス

というようにVAL関数を使えば,数字に戻ります。

14-3 REM文の効率

プログラムをわかり易くするために,プログラムの先頭や,途中にREMARK文を設けます。REMARK文には〔REM〕または〔'〕を用いますが,メモリには,異なる形で記憶されます。

```
10 REM Test
MON
]C60
]D6A4,6A7
06A4 D8 27 E7 27                                                    リ'◤'
]D27D8,27E7
27D8 0F 00 0A 00 01 00 52 45 4D 20 54 65 73 74 00 00          REM Test
]^B
Ok
```

リンクポインタ　行番号10　スペース1コ　REM

REMだと,上に示すように,4バイト使いますが,'だと次に示すように,2バイトですみます。

```
10 ' Test
MON
]C60
]D6A4,6A7
06A4 D8 27 E5 27                                                    リ'◣'
]D27D8,27E5
27D8 0D 00 0A 00 01 00 27 20 54 65 73 74 00 00                  ' Test
]^B
Ok
```

リンクポインタ　行番号10　スペース1コ　'

しかし，さっきはなぜ中間コードにREMという文字で入っていたのでしょう。このREMを別の文字に変えてみましょう。

モニタで直接書き直して，ABCにしてみました。リストをとってもREMのかわりにABCと出ます。RUNをしても，Syntax　errorにはなりません。完全にREM文の働きをしています。

```
10 REM Test
MON
h]C60
h]D6A4,6A7
06A4 D8 27 E7 27                                      ﾘ'◤'
h]D27D8,27E7
27D8 0F 00 0A 00 01 00 52 45 4D 20 54 65 73 74 00 00        REM Test
h]S27DE
27DE 52-41 45-42 4D-43
h]D27D8,27E7
27D8 0F 00 0A 00 01 00 41 42 43 20 54 65 73 74 00 00        ABC Test
h]^B
Ok
list
10 ABC Test
Ok
run
Ok
```

実は，REMの中間コードは，00なのです。インタープリタは，00をステートメントとして抽出した場合REM文として処理し，実行をすぐ次の行に移してしまいます。

したがって，今のように手で書き直せば，REMの中間コードは，00の1バイトしか，消費しないことになります。

次のように，00のあとのREMをTest　OKに変えてもよいのです。

```
10 Test Ok
```

ただし，ここで行を修正したり，この行の上で⏎キーをおすと，REM文ではなくなりますから注意して下さい。

14-4　エラーメッセージをすべて表示するには

エラーを起こしたときに表示されるエラーメッセージは，文字列データとして，ROMの中に格納されています。

ダンプリストでみてみると，

ROM内エラーメッセージの格納され方

セグメント　E800H

```
3A50 01 10 4E 45 58 54 20 77 69 74 68 6F 75 74 20 46     NEXT without F
3A60 4F 52 02 0C 53 79 6E 74 61 78 20 65 72 72 6F 72    OR  Syntax error
3A70 03 14 52 45 54 55 52 4E 20 77 69 74 68 6F 75 74     RETURN without
3A80 20 47 4F 53 55 42 04 0B 4F 75 74 20 6F 66 20 44     GOSUB  Out of D
3A90 41 54 41 05 15 49 6C 6C 65 67 61 6C 20 66 75 6E    ATA  Illegal fun
3AA0 63 74 69 6F 6E 20 63 61 6C 6C 06 08 4F 76 65 72    ction call  Over
3AB0 66 6C 6F 77 07 0D 4F 75 74 20 6F 66 20 6D 65 6D    flow  Out of mem
3AC0 6F 72 79 08 15 55 6E 64 65 66 69 6E 65 64 20 6C    ory  Undefined l
3AD0 69 6E 65 20 6E 75 6D 62 65 72 09 16 53 75 62 73    ine number  Subs
3AE0 63 72 69 70 74 20 6F 75 74 20 6F 66 20 72 61 6E    cript out of ran
3AF0 67 65 0A 14 44 75 70 6C 69 63 61 74 65 20 44 65    ge  Duplicate De
3B00 66 69 6E 69 74 69 6F 6E 0B 10 44 69 76 69 73 69    finition  Divisi
3B10 6F 6E 20 62 79 20 5A 65 72 6F 0C 0E 49 6C 6C 65    on by Zero  Ille
3B20 67 61 6C 20 64 69 72 65 63 74 0D 0D 54 79 70 65    gal direct  Type
3B30 20 6D 69 73 6D 61 74 63 68 0E 13 4F 75 74 20 6F     mismatch  Out o
3B40 66 20 73 74 72 69 6E 67 20 73 70 61 63 65 0F 0F    f string space
3B50 53 74 72 69 6E 67 20 74 6F 6F 20 6C 6F 6E 67 10    String too long
3B60 1A 53 74 72 69 6E 67 20 66 6F 72 6D 75 6C 61 20     String formula
3B70 74 6F 6F 20 63 6F 6D 70 6C 65 78 11 0E 43 61 6E    too complex  Can
3B80 27 74 20 43 6F 6E 74 69 6E 75 65 12 17 55 6E 64    't Continue  Und
3B90 65 66 69 6E 65 64 20 75 73 65 72 20 66 75 6E 63    efined user func
3BA0 74 69 6F 6E 13 09 4E 6F 20 52 45 53 55 4D 45 14    tion  No RESUME
3BB0 14 52 45 53 55 4D 45 20 77 69 74 68 6F 75 74 20     RESUME without
3BC0 65 72 72 6F 72 15 11 55 6E 70 72 69 6E 74 61 62    error  Unprintab
3BD0 6C 65 20 65 72 72 6F 72 16 0F 4D 69 73 73 69 6E    le error  Missin
3BE0 67 20 6F 70 65 72 61 6E 64 17 14 4C 69 6E 65 20    g operand  Line
3BF0 62 75 66 66 65 72 20 6F 76 65 72 66 6C 6F 77 1B    buffer overflow
3C00 0F 54 61 70 65 20 72 65 61 64 20 65 72 72 6F 72     Tape read error
3C10 1D 12 57 48 49 4C 45 20 77 69 74 68 6F 75 74 20     WHILE without
3C20 57 45 4E 44 1E 12 57 45 4E 44 20 77 69 74 68 6F    WEND  WEND witho
3C30 75 74 20 57 48 49 4C 45 1A 10 46 4F 52 20 77 69    ut WHILE  FOR wi
3C40 74 68 6F 75 74 20 4E 45 58 54 1F 0F 44 75 70 6C    thout NEXT  Dupl
3C50 69 63 61 74 65 20 6C 61 62 65 6C 20 0F 55 6E 64    icate label  Und
3C60 65 66 69 6E 65 64 20 6C 61 62 65 6C 21 15 46 65    efined label! Fe
3C70 61 74 75 72 65 20 6E 6F 74 20 61 76 61 69 6C 61    ature not availa
3C80 62 6C 65 32 0E 46 49 45 4C 44 20 6F 76 65 72 66    ble2 FIELD overf
3C90 6C 6F 77 34 0F 42 61 64 20 66 69 6C 65 20 6E 75    low4 Bad file nu
3CA0 6D 62 65 72 35 0E 46 69 6C 65 20 6E 6F 74 20 66    mber5 File not f
3CB0 6F 75 6E 64 33 0E 49 6E 74 65 72 6E 61 6C 20 65    ound3 Internal e
3CC0 72 72 6F 72 36 11 46 69 6C 65 20 61 6C 72 65 61    rror6 File alrea
3CD0 64 79 20 6F 70 65 6E 40 0E 44 69 73 6B 20 49 2F    dy open@ Disk I/
3CE0 4F 20 65 72 72 6F 72 41 13 46 69 6C 65 20 61 6C    O errorA File al
3CF0 72 65 61 64 79 20 65 78 69 73 74 73 37 0E 49 6E    ready exists7 In
3D00 70 75 74 20 70 61 73 74 20 65 6E 64 39 18 44 69    put past end9 Di
3D10 72 65 63 74 20 73 74 61 74 65 6D 65 6E 74 20 69    rect statement i
3D20 6E 20 66 69 6C 65 38 0D 42 61 64 20 66 69 6C 65    n file8 Bad file
3D30 20 6E 61 6D 65 45 14 42 61 64 20 61 6C 6C 6F 63     nameE Bad alloc
3D40 61 74 69 6F 6E 20 74 61 62 6C 65 46 10 42 61 64    ation tableF Bad
3D50 20 64 72 69 76 65 20 6E 75 6D 62 65 72 47 10 42     drive numberG B
3D60 61 64 20 74 72 61 63 6B 2F 73 65 63 74 6F 72 49    ad track/sectorI
3D70 13 52 65 6E 61 6D 65 20 61 63 72 6F 73 73 20 64     Rename across d
3D80 69 73 6B 73 48 0E 44 65 6C 65 74 65 64 20 72 65    isksH Deleted re
3D90 63 6F 72 64 44 09 44 69 73 6B 20 66 75 6C 6C 3B    cordD Disk full;
3DA0 13 53 65 71 75 65 6E 74 69 61 6C 20 49 2F 4F 20     Sequential I/O
3DB0 6F 6E 6C 79 3A 14 53 65 71 75 65 6E 74 69 61 6C    only: Sequential
3DC0 20 61 66 74 65 72 20 50 55 54 3C 0D 46 69 6C 65     after PUT< File
```

```
3DD0 20 6E 6F 74 20 6F 70 65 6E 3D 14 46 69 6C 65 20    not open= File
3DE0 77 72 69 74 65 20 70 72 6F 74 65 63 74 65 64 3E   write protected>
3DF0 0C 44 69 73 6B 20 6F 66 66 6C 69 6E 65 4A 11 49    Disk offlineJ I
3E00 6C 6C 65 67 61 6C 20 6F 70 65 72 61 74 69 6F 6E   llegal operation
```

というふうに，アスキーコードで格納されていて，各文字列の先頭に，エラー番号と文字列の長さが入っています。エンドマークは，このエラー番号と文字列の長さ＝0となっています。

ここに示したダンプリストのアドレスは，ROMのバージョンによって，異なります。次に示すプログラムは，PC-9801では，どのマシンでも，エラーメッセージのリストを表示することができます。

エラーメッセージディスプレイ

```
0 'SAVE"ERROR.DSP"
100  WIDTH 80,25
110  OPEN "SCRN:" AS #1
120 '
130  DEF FNPK(X)=PEEK(X)+PEEK(X+1)*256
140  DEF SEG=0
150 '
160   O1=FNPK(&H310):S1=FNPK(&H312)
170  DEF SEG=S1
180   O2=FNPK(O1+7):O3=FNPK(O2+140)
190   AD=O3+&HCF
200 '
210  NO=PEEK(AD):LN=PEEK(AD+1):AD=AD+2
220   IF NO=0 AND LN=0 THEN 300      :' FETCH END MARK !
230    PRINT #1,USING" ## ･･･ ";NO; :' ERROR No.
240     FOR I=AD TO AD+LN-1
250       PRINT #1,CHR$(PEEK(I));
260     NEXT
270    AD=AD+LN:PRINT #1,""
280  GOTO 210
290 '
300 END
```

プリンターに出力するときは，110行のファイルディスクリプタSCRN：をLPT1：に変えて下さい。なお，PC-9801Fではディスク関連のエラーメッセージは，セグメントFC00H，オフセット366Hから格納されています。

《実行例》

```
 1 ･･･ NEXT without FOR
 2 ･･･ Syntax error
 3 ･･･ RETURN without GOSUB
 4 ･･･ Out of DATA
 5 ･･･ Illegal function call
 6 ･･･ Overflow
 7 ･･･ Out of memory
```

```
 8 ··· Undefined line number
 9 ··· Subscript out of range
10 ··· Duplicate Definition
11 ··· Division by Zero
12 ··· Illegal direct
13 ··· Type mismatch
14 ··· Out of string space
15 ··· String too long
16 ··· String formula too complex
17 ··· Can't Continue
18 ··· Undefined user function
19 ··· No RESUME
20 ··· RESUME without error
21 ··· Unprintable error
22 ··· Missing operand
23 ··· Line buffer overflow
27 ··· Tape read error
29 ··· WHILE without WEND
30 ··· WEND without WHILE
26 ··· FOR without NEXT
31 ··· Duplicate label
32 ··· Undefined label
33 ··· Feature not available
50 ··· FIELD overflow
52 ··· Bad file number
53 ··· File not found
51 ··· Internal error
54 ··· File already open
64 ··· Disk I/O error
65 ··· File already exists
55 ··· Input past end
57 ··· Direct statement in file
56 ··· Bad file name
69 ··· Bad allocation table
70 ··· Bad drive number
71 ··· Bad track/sector
73 ··· Rename across disks
72 ··· Deleted record
68 ··· Disk full
59 ··· Sequential I/O only
58 ··· Sequential after PUT
60 ··· File not open
61 ··· File write protected
62 ··· Disk offline
74 ··· Illegal operation
```

14-5 マシン語でエラーメッセージを表示

8086のアセンブリ言語でプログラミングする際，BASICのエラーメッセージを利用したい場合があります。そこで，アセンブラ・レベルでのエラーメッセージ出力法を考えてみましょう。次のプログラムがそれです。これはエラーコードを指定してコールすればOKです。対応するエラーメッセージがない場合には，"？"が表示されます。

アセンブラで使用するには次のようにします。これは，シンタックスエラーを表示するものです。

```
MOV    AL, [02] ←── エラーコード
CALL   0005H   (Syntax error)
```

これはBASICからでもコールできます。

```
DEF  SEG=&H1F00
ER=0 : E%=[2] ←── エラーコード
CALL   ER(E%)
```

ただし，2FHと43HのRETをIRETに変える必要があります。

PC-9801Fの場合は，3A50Hを3B28Hに変更し，またディスクのエラーメッセージを表示するにはセグメントFC00H，オフセット366Hとして下さい。

BASICでエラーメッセージをコード順に出すには次のようにします。

エラーメッセージをコード順に出力

```
1  save "errprn.bas"
10 DEF SEG=&H1F00
20 DEFINT E : ER=0
30 FOR E=1 TO 74
35 PRINT USING " ## ... ";E;
40  CALL ER(E)
50 NEXT
```

```
                              ;************************
                              ;* ERROR MESSAGE PRINT *
                              ;************************
                              ;
0000 C437                     START:  LES SI,[BX]      ; GET PARA FROM BASIC
0002 268A04                           MOV AL,ES:[SI]   ; AL=ERROR CODE
                              ;
0005 50                       ERR:    PUSH AX
0006 B86000                           MOV AX,60H
0009 8ED8                             MOV DS,AX
000B B800E8                           MOV AX,[0E800H]  ; ROM SEGMENT
000E 8EC0                             MOV ES,AX
0010 B93400                           MOV CX,34H       ; NO.OF ERROR MESSAGES
0013 BB503A                           MOV BX,[3A50H]   ; ERROR MESSAGE ADDRESS
0016 58                               POP AX           ; 3B28H FOR PC-9801F
0017 B600                             MOV DH,0
                              ;
0019 263A07                   SER:    CMP AL,ES:[BX]
001C 7412           0030              JZ FIND          ; ERROR CODE MATCH
001E 43                               INC BX
001F 268A17                           MOV DL,ES:[BX]
0022 43                               INC BX
0023 03DA                             ADD BX,DX        ; SKIP ERROR MASSAGE
```

```
0025 E2F2        0019               LOOP SER
                        ;
0027 B03F                           MOV AL,'?'        ; ? DISPLAY
0029 E82300      004F               CALL DISP
002C E81500      0044               CALL CRLF
002F C3                             RET               ; IRET FOR BASIC
                        ;
0030 43                 FIND:       INC BX
0031 B500                           MOV CH,0
0033 268A0F                         MOV CL,ES:[BX]    ; CX=LEN(MESSAGE)
0036 43                             INC BX
                        ;
0037 268A07             LP:         MOV AL,ES:[BX]    ; MESSAGE PRINT
003A E81200      004F               CALL DISP
003D 43                             INC BX
003E E2F7        0037               LOOP LP
0040 E80100      0044               CALL CRLF
0043 C3                             RET               ; IRET FOR BASIC
                        ;
0044 B00D               CRLF:       MOV AL,0DH        ; CR/LF PRINT
0046 E80600      004F               CALL DISP
0049 B00A                           MOV AL,0AH
004B E80100      004F               CALL DISP
004E C3                             RET
                        ;
004F BF3D00             DISP:       MOV DI,3DH        ; ONE CHR PRINT
0052 CDC4                           INT 0C4H
0054 C3                             RET
                        ;
                                    END
```

出力結果

```
 1 ... NEXT without FOR
 2 ... Syntax error
 3 ... RETURN without GOSUB
 4 ... Out of DATA
 5 ... Illegal function call
 6 ... Overflow
 7 ... Out of memory
 8 ... Undefined line number
 9 ... Subscript out of range
10 ... Duplicate Definition
11 ... Division by Zero
12 ... Illegal direct
13 ... Type mismatch
14 ... Out of string space
15 ... String too long
16 ... String formula too complex
17 ... Can't Continue
18 ... Undefined user function
19 ... No RESUME
20 ... RESUME without error
21 ... Unprintable error
22 ... Missing operand
23 ... Line buffer overflow
24 ... ?
25 ... ?
```

```
26 ... FOR without NEXT
27 ... Tape read error
28 ... ?
29 ... WHILE without WEND
30 ... WEND without WHILE
31 ... Duplicate label
32 ... Undefined label
33 ... Feature not available
34 ... ?
35 ... ?
36 ... ?
37 ... ?
38 ... ?
39 ... ?
40 ... ?
41 ... ?
42 ... ?
43 ... ?
44 ... ?
45 ... ?
46 ... ?
47 ... ?
48 ... ?
49 ... ?
50 ... FIELD overflow
51 ... Internal error
52 ... Bad file number
53 ... File not found
54 ... File already open
55 ... Input past end
56 ... Bad file name
57 ... Direct statement in file
58 ... Sequential after PUT
59 ... Sequential I/O only
60 ... File not open
61 ... File write protected
62 ... Disk offline
63 ... ?
64 ... Disk I/O error
65 ... File already exists
66 ... ?
67 ... ?
68 ... Disk full
69 ... Bad allocation table
70 ... Bad drive number
71 ... Bad track/sector
72 ... Deleted record
73 ... Rename across disks
74 ... Illegal operation
```

14-6 未使用コマンドを使用する

N_{88}-BASIC(86)では，キーワードとして中間コードが割りつけてあっても，使用されていないものがあります。

例えば，ディスクBASICではなくて，ROM-BASICで使っているときのディスク関係のコマンドやGP-IBインターフェースを使っていないときのGP-IB制御関係のコマンドです。ディスク関係の

コマンドはよくご存知だろうし，PC-9801の場合ディスクを使うことを前提として作られているので，説明するまでもないと思いますが，GP-IB関係は制御や測定等に使われるので，使われていない方も多いのではないかと思います。

GP-IB関係のコマンドは，以下のように割にたくさんあります。

ステートメント	中間コード
SRQ	E7
CMD	E8
IRESET	E9
ISET	EA
POLL	EB
RBYTE	EC
WBYTE	ED

関数	中間コード
IEEE	FF FE
STATUS	FF FF

GP-IB 関係のコマンド

これらのキーワードは，しっかりROMに焼き付けられています。GP-IBを使うまでは，上記ステートメントを使用すると，

```
Feature not available
```

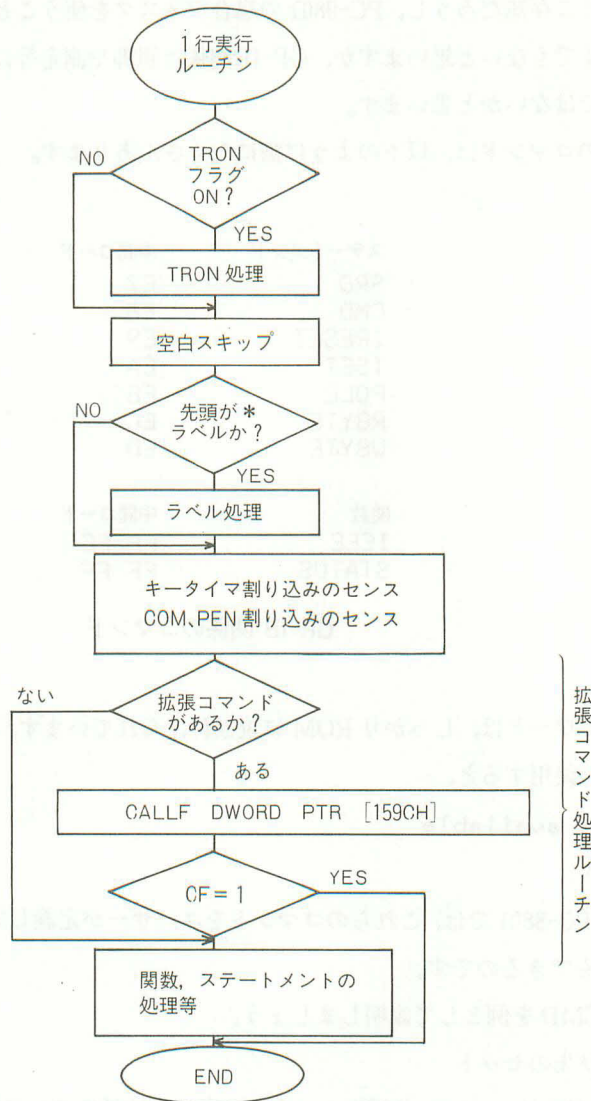
と表示されます。

PC-8001やPC-8801では，これらのコマンドをユーザーが定義して使うことができましたが，実は，PC-9801もできるのです。

ここでは，CMDを例として説明しましょう。

● フラグ飛び先のセット

N_{88}-BASIC(86)は，コマンド解析ルーチンの中で，拡張コマンド使用時のフラグをみて，RAM上のアドレスをFAR CALL（セグメント間コール）をしているのです。

次に，1行実行ルーチンの概略のフローチャートを示します。

上記フローチャートで拡張コマンドがあるかの判断は，

セグメント・ベース　　　６０Ｈ

オフセットアドレス　　　１５９３Ｈ

の内容をみて，

0…………拡張コマンドを使用していない。

0以外……拡張コマンドを使用している。

となっています。

したがって，CMDの使い方は，

1593H	0以外の値をセットする
159C, DH	CMD処理ルーチンのオフセットアドレス
159E, FH	CMD処理ルーチンのセグメントベースアドレス

ということになります。

ただし，これは，ステートメントの処理ルーチンであって，関数処理ルーチンで，拡張コマンドのフラグ（1593 H）をみて，

CALL　DWORD　PTR　[15A0H]

をしている部分がありますので，

15A0, 1H	RETF命令（コード CBH）のあるオフセットアドレス
15A2, 3H	RETF命令（コード CBH）のあるセグメントアドレス

としなければなりません。15A0Hは，IEEE，STATUSの場合コールされます。

● レジスタの保存

関係ないコマンド（この場合CMD以外のコマンド）がやってきたときは，

○キャリーフラグをクリアする（CLC）。

○AL，SIレジスタを保存する。

として，すぐに，RETFを実行します。

目的のCMD，中間コードE8Hがきたときは，目的・処理を実行しますが，

○DSレジスタは保存すること。

○処理がおわって，RETFでリダーンするとき，キャリーフラグをセット（STC）すること。

として下さい。

● CMDルーチンにとび込んできたときの状態

CMDルーチンに飛び込んできたときは，以下のようになっています。

```
10 CMD CLEAR KEY
    ↑    ↖
[6EA,BH] SIレジスタ
```

６Ｅ８，９Ｈ………実行中の行の先頭オフセットアドレス

６ＥＡ，ＢＨ………ＣＭＤ

実際のテキストのダンプリストでは，

```
                    27DD
                     ↓
27D8 0B 00 0A 00 01 E8 01 88 01 A9 00 00 00 00 00 00
27E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

となっています。

Ｅ８Ｈ………ＣＭＤ

８８Ｈ………ＣＬＥＡＲ

Ａ９Ｈ………ＫＥＹ

の中間コードです。

● 次の〝：〞か文のEND MARKの00のあるオフセットアドレスを〔6EAH，6EBH〕にセットして，RETFすること。

となります。

以上を参考にして，未使用コマンドをユーザーで定義して使って下さい。

次に，そのサンプル例を示します。

「第５章 キー入力」で紹介したファンクションキーのイニシャライズ・コマンド

ＣＭＤ　ＣＬＥＡＲ　ＫＥＹ

及び，グラフィック画面で紹介しましたカラーパレットイニシャライズコマンド

ＣＭＤ　ＣＬＥＡＲ　ＣＯＬＯＲ

を組み込んでみました。次ページにソースリストを示しますので参考にして下さい。

```
                        ;
                        ;   SAMPLE OF WAY TO USE ´CMD´
                        ;
                        ;  NEW COMMAND CREATED FLLOWING ...
                        ;
                        ;       CMD CLEAR KEY     INITIALIZE FUNCTION KEYS.
                        ;       CMD CLEAR COLOR  INITIALIZE COLOR PALLET.
                        ;
  00E8                         CMD     EQU      0E8H
  0088                         CLEAR   EQU      88H
  00A9                         KEY     EQU      0A9H
  008E                         COLOR   EQU      8EH
                        ;
                               CSEG
                               ORG     0
                        ;
                        ;   ANALIZE COMMAND
                        ;
0000 3CE8                      CMP     AL,CMD   ; IF NOT CMD
0002 F8                        CLC              ;   THEN CF=0 : END.
0003 7544      0049            JNE     CMD_END
                        ;
0005 E84800    0050            CALL    GET_TOKEN        ; GET TOKEN IN BL
0008 80FB88                    CMP     BL,CLEAR         ; CLEAR ?
000B 753D      004A            JNE     ERROR_END        ; NO THEN ERROR.
000D E84000    0050            CALL    GET_TOKEN        ; GET NEXT TOKEN IN BL
0010 80FBA9                    CMP     BL,KEY           ; KEY ?
0013 7545      005A            JNE     CHECK_COLOR      ; NO THEN CHECK COLOR
                        ;
0015 A1EA06                    MOV     AX,EXEC_TXT      ; SET POINTER TO
0018 A3E806                    MOV     EXEC_TXT_COPY,AX;   NEXT STATEMENT TOP
                        ;
001B 1E                        PUSH    DS  ; MUST BE SAVED DS RESISTER
                        ;
                        ;    INITIALIZE FUNCTIONS KEYS
                            ;
001C 33C0                           XOR     AX,AX
001E 8ED8                           MOV     DS,AX
0020 C51E1003                       LDS     BX,DWORD PTR VECT
                            ;
0024 83C307                         ADD     BX,7
0027 8B07                           MOV     AX,[BX]
0029 052A00                         ADD     AX,2AH
002C 8BD8                           MOV     BX,AX
002E 8B1F                           MOV     BX,[BX]
0030 B88006                         MOV     AX,680H ; FNKEY FIRST DATA.
                            CMDKEY10:
0033 3B07                           CMP     AX,[BX]
0035 7403       003A                JE      CMDKEY20
0037 43                             INC     BX
0038 EBF9           0033            JMPS    CMDKEY10
                            CMDKEY20:
003A FC                             CLD
003B 16                             PUSH    SS
003C 07                             POP     ES
003D B9B400                         MOV     CX,0B4H
0040 BF7803                         MOV     DI,378H
0043 8BF3                           MOV     SI,BX
0045 F3A4                           REP     MOVSB
                            ;
                            ;
```

```
0047 1F                     POP     DS          ; LOAD DS RESISTER
                   CF1_END:
0048 F9                     STC                 ; CF=1
                   CMD_END:
0049 CB                     RETF                ; RETURN TO BASIC
                   ;
                   ERROR_END:
004A BF0100                 MOV     DI,1        ; DISPLAY 'SYNTAX ERROR'
004D CDC4                   INT     0C4H
004F CB                     RETF
                   ;
                   GET_TOKEN:
0050 8936EA06               MOV     EXEC_TXT,SI
0054 BF0D00                 MOV     DI,13              ; GET TOKEN
0057 CDC4                   INT     0C4H
0059 C3                     RET
                   ;
                   CHECK_COLOR:
005A 80FB8E                 CMP     BL,COLOR           ; COLOR ?
005D 75EB          004A     JNE     ERROR_END          ; NO THEN ERROR.
                   ;
005F A1EA06             MOV     AX,EXEC_TXT        ; SET POINTER TO
0062 A3E806             MOV     EXEC_TXT_COPY,AX;   NEXT STATEMENT TOP
                ;
                ;   INITIALIZE COLOR PALLET
                ;
0065 BB4406             MOV     BX,644H
0068 B86745             MOV     AX,4567H
006B 8907               MOV     [BX],AX
006D B82301             MOV     AX,0123H
0070 43                 INC     BX
0071 43                 INC     BX
0072 8907               MOV     [BX],AX
0074 BB4006             MOV     BX,640H
0077 B443               MOV     AH,43H
0079 CD18               INT     18H
007B EBCB     0048      JMPS    CF1_END
                ;
                ;   DEFINE BASIC DATA AREA
                ;
 0060                   DSEG    60H
                ;
                        ORG     6E8H

06E8       EXEC_TXT_COPY  RW     1          ; POINTER OF EXEC TEXT
06EA       EXEC_TXT       RW     1          ; POINTER OF EXEC TEXT
           ;
                     ORG    310H
0310                 VECT   RS     4        ; INT VECTOR TABLE
           ;
                     END
```

N_{88}-BASIC (86) のモニタ (MON) では，ディスク BASIC 時に簡易アセンブラ・ディスアセンブラがついていますが，このアセンブラでは，RETF や CALLF などのセグメント間命令がサポートされていません。

RETF の方は，1バイトの CBH ですので，S コマンドで直接，書けばよいでしょう。

次に示すプログラムは BASIC で書かれていますが,前出のサンプルプログラムのマシンコードを DATA 文で拾ったものです。この BASIC プログラムを入力して走らせれば，そのときから，

```
CMD CLEAR KEY
CMD CLEAR COLOR
```

という2つのコマンドが使えます。ユーザーマシン語領域を使っていますので，他のマシン語プログラムといっしょに使うときは注意して下さい。

CMD を使ったファンクションキー及びカラーパレットのイニシャライズコマンド

```
0 'SAVE"CMDKEY.BAS"
100 '
110 '   CMD CLEAR KEY
120 '
130  DEF SEG=&H60
140  POKE &H1593,0 :' EXPAND COMMAND FLAG OFF
150 '
160  CLEAR ,&H1FF0:DEF SEG=&H1FF0
170 '
180  FOR I=0 TO &H7C
190    READ A$:POKE I,VAL("&H"+A$)
200  NEXT
210 ' FOR CMD
220  DEF SEG=&H60
230  POKE &H159C,0    :POKE &H159D,0    :' OFFSET
240  POKE &H159E,&HF0:POKE &H159F,&H1F:' SEGMENT
250  POKE &H1593,1      :' SET EXPAND COMMAND FLAG
260 ' FOR IEEE AND STATUS
270  POKE &H15A0,&H75:POKE &H15A1,0    :' OFFSET
280  POKE &H15A2,&HF0:POKE &H15A3,&H1F:' SEGMENT
290 '
300  BEEP 1:PRINT"CREATE 'CMD CLEAR KEY'":BEEP 0
310 '
320 END
330 '
340 DATA 3C,E8,F8,75,44,E8,48,00,80,FB,88,75,3D,E8,40,00
350 DATA 80,FB,A9,75,45,A1,EA,06,A3,E8,06,1E,33,C0,8E,D8
360 DATA C5,1E,10,03,83,C3,07,8B,07,05,2A,00,8B,D8,8B,1F
370 DATA B8,80,06,3B,07,74,03,43,EB,F9,FC,16,07,B9,B4,00
380 DATA BF,78,03,8B,F3,F3,A4,1F,F9,CB,BF,01,00,CD,C4,CB
390 DATA 89,36,EA,06,BF,0D,00,CD,C4,C3,80,FB,8E,75,EB,A1
400 DATA EA,06,A3,E8,06,BB,44,06,B8,67,45,89,07,B8,23,01
410 DATA 43,43,89,07,BB,40,06,B4,43,CD,18,EB,CB
```

参考までに，マシン・コードを BASIC の DATA 文に吸い上げる簡単なプログラムを示します。RUN すると，スタートアドレスとエンドアドレスをきいてくるので，16 進表記で入力して下さい。

すると，画面に10000行から10行おきに，DATA文になったマシンコードが表示されますので，一画面以内のところでSTOPして，先頭の行番号にもっていって，↵キーをおしていって下さい。

　　　DELETE　100－260

として，DATA文作成プログラムを消去して，DATA文だけSAVEすればよいでしょう。

　　　LN………最初の行番号

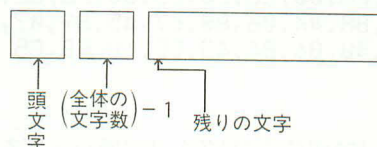
となっていますので，100行の10000を変えると最初の行番号が変わり，増分は，240行の10をかえて下さい。

セグメントアドレスは，110行で決めています。

DATA文作成プログラム

```
0 'SAVE"DATA.CRT"
100  LN=10000
110  DEF SEG=&H1F00
120  DEF FNHX$(X)=RIGHT$("0"+HEX$(X),2)
130  DEF FNHXW$(X)=RIGHT$("000"+HEX$(X),4)
140 '
150  INPUT "START ADDRESS=",S$:S=VAL("&H"+S$)
160  INPUT "END   ADDRESS=",E$:E=VAL("&H"+E$)
170  IF S>E THEN END
180 '
190   FOR I=S TO E STEP 16
200    PRINT STR$(LN)" DATA ";
210     FOR J=0 TO 15
220      PRINT FNHX$(PEEK(I+J));:IF J<>15 THEN PRINT ",";
                                      ELSE PRINT ":'"FNHXW$(I)"H";
230     NEXT:PRINT
240      LN=LN+10
250   NEXT
260  END
```

14-7　新しいコマンドを作る

前節で，未使用コマンドを使用する方法を述べましたが，新しいコマンドを作ることができるのです。つまり，キーワードに登録されている以外のコマンドを作ることができます。キーワード以外の文字列は，ダブルクオテーションやREMの外では，変数名として，テキストに置かれますので，

という形になっています。

前節のフックアドレスを利用すると，RAM 上の拡張コマンドの処理ルーチンに入ったときに，例えばINIKEY という新しいコマンドを作ったとしますと，

```
I  05  N  I  K  E  Y
   ↑
   SI
```

AL＝先頭の〝I〟が入っている。　SIは05をさしている。

という状態になっていますので，もし，AL レジスタにIが入っているとき，そのあとに05 NIKEY が連続して続いているときに，INIKEY 処理ルーチンに入り，それ以外は，キャリーフラグをクリアしてすぐにRETF するようにします。

レジスタの保存等は，前節と同じです。

次に，新しいコマンド INIKEY に，ファンクションキーイニシャライズを加えた例を示しておきます。

```
                                ;
                                ;     FREE COMMAND CREATED
                                ;
                                ;   THE NEW COMMAND 'INIKEY'
                                ;
0000 50                                 PUSH    AX        ; PROTECT AX,SI
0001 56                                 PUSH    SI
0002 3C49                               CMP     AL,'I'    ; IF NOT 'I'
                                                          ;   THEN END.
0004 755D           0063                JNE     COM_END
0006 AC                                 LODSB
0007 3C05                               CMP     AL,5      ; NUMBER OF LETTERS=5 ?
0009 7558           0063                JNE     COM_END
000B AC                                 LODSB
000C 3C4E                               CMP     AL,'N'    ; N ?
000E 7553           0063                JNE     COM_END
0010 AC                                 LODSB
0011 3C49                               CMP     AL,'I'    ; I ?
0013 754E           0063                JNE     COM_END
0015 AC                                 LODSB
0016 3C4B                               CMP     AL,'K'    ; K ?
0018 7549           0063                JNE     COM_END
001A AC                                 LODSB
001B 3C45                               CMP     AL,'E'    ; E ?
001D 7544           0063                JNE     COM_END
001F AC                                 LODSB
0020 3C59                               CMP     AL,'Y'    ; Y ?
0022 753F           0063                JNE     COM_END
                                ;
                                LOOP:
0024 AC                                 LODSB
0025 3C007404       002D                CMP AL,0   ! JE   LOOP_END
0029 3C0B72F7       0024                CMP AL,0BH ! JB   LOOP
```

```
                           LOOP_END:
002D 4E                            DEC      SI
002E 8936E806                      MOV      EXEC_TXT_COPY,SI

                           ;
                           ;    INITIALIZE FUNCTION KEYS
                           ;
                           ;         'INIKEY'
                           ;
0032 1E                            PUSH     DS         ; PROTECT DS
0033 33C0                          XOR      AX,AX
0035 8ED8                          MOV      DS,AX
0037 C51E1003                      LDS      BX,DWORD PTR VECT
                           ;
003B 83C307                        ADD      BX,7
003E 8B07                          MOV      AX,[BX]
0040 052A00                        ADD      AX,2AH
0043 8BD8                          MOV      BX,AX
0045 8B1F                          MOV      BX,[BX]
0047 B88006                        MOV      AX,680H ; FNKEY FIRST DATA
                           INIKEY10:
004A 3B07                          CMP      AX,[BX]
004C 7403             0051         JE       INIKEY20
004E 43                            INC      BX
004F EBF9             004A         JMPS     INIKEY10
                           INIKEY20:
0051 FC                            CLD
0052 16                            PUSH     SS
0053 07                            POP      ES
0054 B9B400                        MOV      CX,0B4H
0057 BF7803                        MOV      DI,378H
005A 8BF3                          MOV      SI,BX
005C F3A4                          REP      MOVSB
                           ;
005E 1F                            POP      DS         ; PROTECT DS
                           ;
005F F9                            STC      ; CF=1
                           INI_END:
0060 5E                            POP      SI         ; PROTECT AX,SI
0061 58                            POP      AX
0062 CB                            RETF
                           COM_END:
0063 F8                            CLC
0064 EBFA             0060         JMPS     INI_END
                           ;
                           ;    DEFINE BASIC DATA AREA
                           ;
0060                               DSEG     60H
                           ;
                                   ORG      6E8H
06E8                       EXEC_TXT_COPY    RW         1 ; POINTER OF EXEC TEXT
                           ;
                                   ORG      310H
0310                               VECT     RS         4 ; INT VECTOR TABLE
                           ;
                                   END
```

上記マシンコードをBASICのDATA文で吸い上げました。次のBASICプログラムを入力しRUNすれば，新しいコマンドINIKEYがふえます。

新しいコマンドINIKEYを使ったファンクションキーイニシャライズ

```
0 'SAVE"INIKEY.BAS"
100 '
110 '  CREATE NEW COMMAND 'INIKEY'
120 '
122  DEF SEG=&H60
124  POKE &H1593,0             :' EXPAND COMMAND FLAG OFF
126 '
130  CLEAR ,&H1FF0:DEF SEG=&H1FF0
140 '
150  FOR I=0 TO &H65
160   READ A$:POKE I,VAL("&H"+A$)
170  NEXT
180 '
190  DEF SEG=&H60
200  POKE &H159C,0    :POKE &H159D,0    :' NEW COM OFFSET
210  POKE &H159E,&HF0:POKE &H159F,&H1F:' NEW COM SEGMENT
220 '
230  POKE &H15A0,0:POKE &H15A1,0 :' IEEE & STATUS OFFSET
240  POKE &H15A2,&HF0:POKE &H15A3,&H1F:'
250 '
260  POKE &H1593,1                    :' EXPAND COMMAND FLAG
270 '
280  BEEP 1:PRINT "NOW CREATE 'INIKEY'":BEEP 0
290 '
300 END
10000 DATA 50,56,3C,49,75,5D,AC,3C,05,75,58,AC,3C,4E,75,53
10160 DATA AC,3C,49,75,4E,AC,3C,4B,75,49,AC,3C,45,75,44,AC
10320 DATA 3C,59,75,3F,AC,3C,00,74,04,3C,0B,72,F7,4E,89,36
10480 DATA E8,06,1E,33,C0,8E,D8,C5,1E,10,03,83,C3,07,8B,07
10640 DATA 05,2A,00,8B,D8,8B,1F,B8,80,06,3B,07,74,03,43,EB
10800 DATA F9,FC,16,07,B9,B4,00,BF,78,03,8B,F3,F3,A4,1F,F9
10960 DATA 5E,58,CB,F8,EB,FA
```

ただし，このようにして増やしたコマンドは，BASICインタープリタ内では，変数名扱いを受けますので，

? I N I K E Y

とした場合は，値0をもった変数となります。しかし，変数として代入はできません。即ち，

I N I K E Y＝6

で，INIKEYという変数に6を代入できません。＝6はINIKEY処理ルーチン内での引数となります。今の例では，引数がありませんので，

Syntax　error

になります。

14-8　8086はリセットがかかったら何処へ!?

Z-80など8ビットのシステムでリセットをかけると，0000番地から実行されます。それで，PC-8801などで，

```
mon　　［RET］
＊G0　　［RET］
```

として，ソフト的にリセットかけることができます。

では，8086ではどうなるのでしょう？ 8086はリセット後，次のように各レジスタがセット（リセット）されます。

ＩＰ………０００００Ｈ（命令ポインタ）
ＣＳ………ＦＦＦＦＨ（コードセグメント）
ＤＳ………００００Ｈ
ＳＳ………００００Ｈ
ＥＳ………００００Ｈ
フラグ……すべてクリア

そのため，PC-9801をリセットすると物理アドレスのFFFF0Hから実行を始めるわけです。そこで，FFFF0Hの内容をみてみると，次のようになっています。

EA 00 00 80 FD

これは，セグメント間ジャンプ命令で，セグメントFD80H，オフセット0000Hにジャンプするということです（モニタの逆アセンブラでは正常にとれません）。物理アドレスのFD800Hからは，BIOSが入っており，これを実行後，N_{88}-BASIC（86）が起動されます。

ちなみに，リセットをかけず，ソフト的にOSを起動する法があります。セグメントFD80H，オフセット91EHからは，OSのブートストラップローダー（ディスクシステムの起動プログラム）が入っており，

```
DEF　　SEG=&HFD80
A=&H91E
```

CALL　A

または，

mon　[RET]

h] CFD80　[RET]

h] G91E　[RET]

とすると，N_{88}-Disk BASIC (86)，CP／M-86，MS-DOS などが起動します。

PC-9801のリセットボタンを押すのではなく，ソフト的にリセットをかけることがあれば，以上の方法をご使用ください。ハードウェアリセットよりも数秒速く起動できます。

14-9 INKEY$でカーソル表示

INKEY$関数でキーセンスを行う場合，カーソルが表示されませんので，入力位置などが分からなくなるときがあります。そこで，INKEY$でもカーソルを表示させる方法を紹介します。

カーソル表示プログラム

```
1   ' SAVE "CURSOR.BAS"
100 DEFINT X-Y :A=0
110 DEF SEG=&H1F00
120 FOR I=0 TO &H1E
130  READ D$:D=VAL("&H"+D$)
140  POKE I,D
150 NEXT I
160 X=POS(0):Y=CSRLIN: CALL A(X,Y)
170 A$=INKEY$:IF A$="" THEN 160
180 IF A$=CHR$(13) THEN PRINT
190 PRINT A$;
200 GOTO 160
210 DATA C4,77,04        ' LES SI,04[BX]  ┐
220 DATA 26              ' ES:            ├→; DL=X
230 DATA 8A,14           ' MOV DL,[SI]    ┘
240 DATA C4,37           ' LES SI,[BX]    ┐
250 DATA 26              ' ES:            ├→; DH=Y
260 DATA 8A,34           ' MOV DH,[SI]    ┘
270 DATA B8,A0,00        ' MOV AX,00A0      ; AX=160
280 DATA F6,E6           ' MUL DH           ; AX=160×DH
290 DATA 30,F6           ' XOR DH,DH
300 DATA 00,D2           ' ADD DL,DL        ; DL=DL+DL
310 DATA 01,C2           ' ADD DX,AX        ; DX=DX+AX
                                                 |   └ 160×DH
                                                 DL×2
320 DATA B4,13           ' MOV AH,13 ┐
330 DATA CD,18           ' INT 18    ┘→ CURSOR LOCATE
340 DATA B4,11           ' MOV AH,11 ┐
350 DATA CD,18           ' INT 18    ┘→ CURSOR DISPLAY
360 DATA CF              ' IRET
```

ここでのポイントは，カーソル表示位置を指定して，そこにカーソルを表示させなくてはいけないことです。幸い，ROM 内にそれらのルーチンがあるので利用してみましょう。

まず，X，Yにカーソル位置を入れて，マシン語ルーチンをコールします。カーソル位置指定は次のとおりです。

```
MOV    DX，VADRS    ；カーソル位置
MOV    AH，13H
INT    18H
```

DXに入る値は，CPUから見たVRAM上のアドレスで，

```
VADRS＝160＊Y＋2＊X
```

で求められます。

マシン語ルーチンで，X，Yの値を上の式のとおり変換してDX に入れています。そして，

```
MOV    AH，11H
INT    18H
```

でカーソル表示します。

14-10 高速リスト

LIST コマンドは，人間の目でおってゆけるように，タイミングをとっています。このタイミングは可変になっていて，セグメント 60 H のオフセット 1802 H，1803 H に，そのタイミングを入れるようになっています。

1802 H，1803 H が 0000 H 以外のときは，その値を CX に入れて，

```
LOOP    $
```

でタイミングをとります。0000 H のときは，デフォールト値として 6000 H をとります。

したがって，6000 H より小さいときは，通常より速く，6000 H より大きいときは，通常よりおそい LIST がとれます。

それでは，高速 LIST にしてみましょう。

```
MON
hJC60
hJS1802
```

```
1802 00-01
h]^B
Ok
```

これで，高速リストになります。タイミングとして0001Hとしたのです。
LISTをとってみて下さい！ただしEとFでは、タイミングの値がROM内にありますので、変更することはできません。

14-11 CHR$(13);CHR$(10)とCHR$(13)+CHR$(10)との違い

CRT上にPRINT出力する場合，

```
10 PRINT "ABC";CHR$(13);CHR$(10)
20 PRINT "DEF"
```

とすると，ABCと書いた行とDEFと書いた行はつながってしまいますが，

```
10 PRINT "ABC";CHR$(13)+CHR$(10)
20 PRINT "DEF"
```

とすると，2つの行は分離されます。

この理由は，キャリッジリターン(0DH)とラインフィード(0AH)を1つ1つ出力すると，CRT出力ルーチンは，行の区切りを発生せず，0DHと0AHを一緒に出力すると，行の区切りを発生するように，プログラムしてあるからです。

したがって，INT C4HでROM内ルーチンを呼び出して，CRT出力を行う場合は，0DH，0AHを1つ1つ出力すると，行がすべてつながってしまいます。

行がつながってほしくない場合は，文字出力バッファ（セグメント60Hのオフセット202H）に0DH，0AHを入れて，文字カウンタCXに2を入れて，0DH，0AHを同時に出力しなければなりません。

14-12 OUTPUTとASも変数に使える

ファイルをオープンするときは，

```
OPEN "ファイル名" FOR OUTPUT AS #1
```

というステートメントを使いますが，

OUTPUTとAS

は，中間言語表にありませんから，変数名として格納されています。ということは，変数として使えるのでしょうか？

実際やってみましょう。

```
OUTPUT=12
Ok
AS=OUTPUT*2
Ok
PRINT OUTPUT;AS
 12  24
Ok
```

ちゃんと使えますね！

14-13 キーバッファクリア

「第5章　キー入力」にあるように，N_{88}-BASIC(86)はキーバッファがあり，キーの先行入力が可能です。これはたいへん重宝な機能ですが，プログラム実行中に不要なキー入力を避けたい場合があります。そんなときには，適所適所でキーバッファをクリアすれば問題ありません。その方法を3通り紹介します。

```
①100 IF INKEY$<>"" THEN 100
```

```
②200 WHILE INKEY$<>"":WEND
```

```
③300 DEF SEG=&H1F00
 310 FOR I=0 TO 2:READ D$
 320  D=VAL("&H"+D$)
 330  POKE I,D:NEXT I
 340 KC=0:CALL KC
 350 DATA CD,9E :' INT 9EH
 360 DATA CF    :' IRET
```

①と②はBASICのステートメントを用いたものです。③はROM内ルーチンのインタラプトコール(INT 9EH)を利用したものでBASICからの実行例です。アセンブリ言語でのプログラミングの際に役立てて下さい。

14-14 リアルタイムで時間を表示

PC を使用中に現在の時間がリアルタイムに CRT に表示されていればいいなと思ったことはありませんか？時計が手もとにない。また掛時計も置き時計もない。PC はフルに稼動中で，PRINT TIME$を実行することもできない。そんなときに，次のプログラムを実行しておけば、CRT の右上のすみにリアルタイムに時間が表示されます。

これは，インターバル・タイマ割り込みをセットして、時間を読み，それをテキスト VRAM に書き込んでいます。ソースリストもあわせて掲げておきますので，割り込み処理の参考にして下さい。なお，時間の表示を止めるには，次のステートメントをダイレクトで実行します。

ＤＥＦ　ＳＥＧ＝＆Ｈ１Ｄ００：ＰＯＫＥ　0，＆ＨＣＦ

時間表示プログラム

```
1 'save "TIME.BAS"
100 ' Real Time Time Display
110 ' ---  TM=0:CALL TM ---
120 WIDTH 80,25 : CONSOLE 1,24
130 DEF SEG=&H1D00
140  FOR I=0 TO &H5E : READ D$
150   D=VAL("&H"+D$) : POKE I,D
160  NEXT I
170 TM=0:CALL TM
180 LOCATE 0,1:END
190 DATA FA,06,1E,56,51,53,50,B9,32,00,B4,02,0E,07,BB,00
200 DATA 00,CD,1C,B4,00,0E,07,BB,5F,00,CD,1C,B8,00,A0,8E
210 DATA D8,BE,8C,00,E8,1A,00,B0,3A,88,04,46,46,E8,11,00
220 DATA B0,3A,88,04,46,46,E8,08,00,58,5B,59,5E,1F,07,FB
230 DATA CF,26,8A,67,03,43,8A,C4,24,F0,B1,04,D2,C8,0C,30
240 DATA 88,04,46,46,8A,C4,24,0F,0C,30,88,04,46,46,C3,90
```

時間表示ソースリスト

```
                          ;===================================
                          ; Real Time Time Display
                          ; Calling sequence:
                          ;  DEF SEG=&H1D00:TM=0
                          ;  CALL TM 'Display On
                          ;===================================
                          ;
 1D00                              CSEG 1D00H
                                   ORG 0
                          ;
0000 FA                   GETP:    CLI
0001 06                   INTSET:  PUSH ES          ; Save Main Regs
0002 1E                            PUSH DS
0003 56                            PUSH SI
0004 51                            PUSH CX
0005 53                            PUSH BX
0006 50                            PUSH AX
0007 B93200                        MOV CX,50        ; CX<=50 x 10 msec
000A B402                          MOV AH,02H       ; Interrupt Set
```

```
000C 0E                          PUSH CS
000D 07                          POP ES
000E BB0000                      MOV BX,0         ; ES:BX
0011 CD1C                        INT 1CH          ; User Routine
                          ;-------------------------------------------
                          MAIN:
0013 B400                        MOV AH,00H       ; Time Read
0015 0E                          PUSH CS
0016 07                          POP ES
0017 BB5F00                      MOV BX,OFFSET TBUFF  ; Time Read Buffer
001A CD1C                        INT 1CH
                          ;-------------------------------------------
001C B800A0               TIME:  MOV AX,0A000H
001F 8ED8                        MOV DS,AX        ; DS<=Text V-RAM
0021 BE8C00                      MOV SI,140       ; Locate 70,0
0024 E81A00        0041          CALL DISP        ; HOURE
0027 B03A                        MOV AL,':'
0029 8804                        MOV [SI],AL
002B 46                          INC SI
002C 46                          INC SI
002D E81100        0041          CALL DISP        ; MINUTE
0030 B03A                        MOV AL,':'
0032 8804                        MOV [SI],AL
0034 46                          INC SI
0035 46                          INC SI
0036 E80800        0041          CALL DISP        ; SECOND
0039 58                          POP AX           ; Restore Main Regs
003A 5B                          POP BX
003B 59                          POP CX
003C 5E                          POP SI
003D 1F                          POP DS
003E 07                          POP ES
003F FB                          STI
0040 CF                          IRET             ; Return
                          ;-------------------------------------------
0041 268A6703             DISP:  MOV AH,ES:03H[BX]  ; Time Buffer
0045 43                          INC BX
0046 8AC4                        MOV AL,AH
0048 24F0                        AND AL,0F0H      ; Top 4 bits
004A B104                        MOV CL,4         ; Rotate 4 times
004C D2C8                        ROR AL,CL
004E 0C30                        OR AL,30H
0050 8804                        MOV [SI],AL
0052 46                          INC SI
0053 46                          INC SI
                          ;
0054 8AC4                        MOV AL,AH        ; Under 4 bits
0056 240F                        AND AL,0FH
0058 0C30                        OR AL,30H
005A 8804                        MOV [SI],AL
005C 46                          INC SI
005D 46                          INC SI
005E C3                          RET              ; Return to MAIN
                          ;-------------------------------------------
005F                      TBUFF  RS 6             ; Time Read Buffer
                          ;
                                 END
```

14-15 モニタモードでファンクションキーを使用する

モニタモードに入ると，ファンクションキーが使えなくなります。これは，MON のルーチンが，ファンクションキーのフラグを 20H にしてキー割り込み ON の状態にしているためです。このフラグをモニタに入った後，80H に変更すれば，通常のファンクションキーとして使えるようになります。ただし，キーの内容が1文字置きにしか表示されませんが……。

次のプログラムを実行すると，モニタモードに入ります。そこで，G 0,11 [RET] とすれば，以後ファンクションキーが使えるようになります。例として，アセンブラーで使用するニーモニックを入れています。なお，CTRL-B で BASIC に戻ると，自動的にキーの内容がもとのものに書き換わります。

モニタモードでファンクションキーを使用

```
1 ' save "FUNK.MON"
100 ' = Use F.KEYS in MON mode =
110 WIDTH 80,25
120 DEF SEG=&H60 : DIM F(179)
130 FOR I=0 TO 179
140  F(I)=PEEK(&H378+I)
150 NEXT I
160 DEF SEG=&H1F00
170 FOR I=0 TO 16
180  READ D$ : D=VAL("&H"+D$)
190  POKE I,D
200 NEXT I
210 FOR I=1 TO 10
220   READ D$ : KEY I,D$
230 NEXT I
240 MON
250 DEF SEG=&H60
260 FOR I=0 TO 179
270  POKE &H378+I,F(I)
280 NEXT I : WIDTH 80
290 END
300 ' -- F.KEY ON --
310 DATA 16            ' PUSH  SS
320 DATA 1F            ' POP   DS
330 DATA B9,0A,00      ' MOV   CX,000A
340 DATA B0,80         ' MOV   AL,80
350 DATA BB,78,03      ' MOV   BX,0378
360 DATA 88,07         ' MOV   [BX],AL
370 DATA 83,C3,12      ' ADD   BX,0012
380 DATA E2,F9         ' LOOP  000A
390 ' -- F.KEY DATA --
400 DATA "M O V  ","C A L L  "
410 DATA "P U S H  ","P O P  "
420 DATA "I R E T  ","R E T  "
430 DATA "I N T  ","[ B X ] "
440 DATA "[ S I ] ","[ D I ] "
```

第 15 章　ユーティリティ

15-1　テキストサーチ

15-2　リプレイス

15-3　バリアブルリスト

15-4　バーティカル・ファイルズ

第15章　ユーティリティ

ここでは，BASICプログラムのデバッグのためとディスク関連のユーティリティプログラムを紹介します。

巻末にある「インタープリタ内ルーチンの利用（INT　C4H)」のサンプルプログラムとしても，参考になると思います。

15-1　テキストサーチ

コマンド16(10H〝テキスト内部表現——→ソースイメージ変換〞)を使えば，RAM上にある内部表現のBASICテキスト内のテキストサーチが行えます。

内部表現をソースイメージに変換し，見つけたい文字列とASCIIコードで比較すればよいのです。

このプログラムはUSR文で呼び出します。例えば，IFという文字列を見つけたいときは，

```
A=USR("IF")
```

とします。A$=〝IF〞として，

```
A=USR(A$)
```

としてもよいです。捜したい文字列を含む行がリストアップされます。そのままカーソルをもっていって修正もできます。USR文を使うとき，ユーザー処理ルーチンでBASICに戻る時，

```
XOR	AX, AX
IRET
```

とすれば，左辺の変数は引数の型と一致していなくても，Type mismatchエラーは起こりません。

文字の出力は，コマンド37(25H)の〝カレントデバイスへの出力〞を用いていますので，セグメント60H，オフセット1840Hを3にするとプリンタ，4にするとCRTに出力されます。通常CRTになっています。このルーチンは，LISTコマンドも用いていますので，LLISTをしたすぐあとでは，プリンタに出力されます。

テキストサーチ

```
0 ´SAVE"FIND.BAS"
100 ´
110 ´    CALLING SEQUENCE IS:
120 ´
130 ´          A=USR(" STRINGS TO BE FOUND ")
140 ´
150 ´     CTRL-S  : ESCAPE FIND ROUTINE,RETURN WITH OTHER KEY.
160 ´     STOP    : STOP THE FIND ROUTINE.
170 ´
180  CLEAR ,&H1F00:DEF SEG=&H1F00:DEF USR=0
190  FOR I=0 TO &H1AE
200   READ A$:POKE I,VAL("&H"+A$)
210  NEXT
220 END
10000 DATA 3C,03,74,03,E9,C5,00,36,A1,A4,06,36,A3,DE,06,16:´0000H
10010 DATA 1F,C5,1F,8A,0F,43,8A,07,43,8B,37,0A,C0,75,04,8E:´0010H
10020 DATA DA,EB,02,16,1F,E8,5C,00,16,07,BF,03,02,E8,78,00:´0020H
10030 DATA 73,03,E8,0D,00,E8,30,00,72,05,E8,6B,01,EB,E6,33:´0030H
10040 DATA C0,CF,51,56,1E,16,1F,BE,03,02,AC,0A,C0,74,05,E8:´0040H
10050 DATA 85,00,EB,F6,B8,0D,0A,36,A3,02,02,B9,02,00,BF,25:´0050H
10060 DATA 00,E8,39,01,1F,5E,59,C3,1E,16,1F,36,8B,1E,DE,06:´0060H
10070 DATA 8B,07,03,D8,83,3F,00,74,08,36,89,1E,DE,06,F8,EB:´0070H
10080 DATA 01,F9,1F,C3,56,E8,10,00,BB,03,02,36,8B,36,DE,06:´0080H
10090 DATA BF,10,00,E8,07,01,5E,C3,51,16,07,BF,03,02,33,C0:´0090H
10100 DATA B9,80,00,FC,F3,AB,59,C3,56,51,57,56,51,A6,75,06:´00A0H
10110 DATA FE,C9,75,F9,EB,0F,59,5E,5F,47,26,8A,05,0A,C0,75:´00B0H
10120 DATA E9,59,5E,F8,C3,59,5E,5F,59,5E,F9,C3,BF,03,00,E8:´00C0H
10130 DATA CB,00,CF,00,00,00,00,3C,1B,75,07,2E,C6,06,D3,00:´00D0H
10140 DATA 01,C3,2E,80,3E,D3,00,01,75,23,B4,00,3C,4B,74,49:´00E0H
10150 DATA 2E,88,26,D4,00,2E,88,26,D3,00,36,80,3E,40,18,04:´00F0H
10160 DATA 74,0A,B0,1B,E8,22,00,B0,48,E8,1D,00,C3,2E,80,3E:´0100H
10170 DATA D4,00,01,75,14,2E,80,3E,D5,00,00,75,3C,2C,20,2E:´0110H
10180 DATA A2,D6,00,2E,FE,06,D5,00,C3,51,B9,01,00,36,A2,02:´0120H
10190 DATA 02,BF,25,00,E8,66,00,59,C3,FE,C4,2E,88,26,D4,00:´0130H
10200 DATA 36,80,3E,40,18,04,74,0A,B0,1B,E8,DC,FF,B0,4B,E8:´0140H
10210 DATA D7,FF,2E,C6,06,D3,00,00,C3,8A,E0,2E,A0,D6,00,E8:´0150H
10220 DATA 07,00,2E,C6,06,D5,00,00,C3,51,1E,06,16,1F,36,80:´0160H
10230 DATA 3E,40,18,04,75,16,36,A3,00,00,36,A3,02,00,36,8E:´0170H
10240 DATA 06,12,14,B9,04,00,CD,89,07,1F,59,C3,04,20,36,A3:´0180H
10250 DATA 02,02,BF,25,00,B9,02,00,E8,02,00,EB,EB,1E,56,51:´0190H
10260 DATA 16,1F,CD,C4,59,5E,1F,C3,BF,4B,00,E8,EF,FF,C3   :´01A0H
```

次にマシン語部分のソースリストを掲げます。

```
                          ;
                          ;     THIS COMMAND FIND STRINGS
                          ;
                          ;     CALLING SEQUENCE
                          ;
                          ;         A=USR(" STRINGS TO BE FOUND ")
                          ;
                          ;
                                    SSEG
                          ;
                                    ORG     0000H
0000                      PRNTBUF   RS      512     ; PRINT BUFFER
                          ;
                                    ORG     203H
0203                      TXTBUF    RS      256     ; ASCII TEXT BUFFER
                          ;
```

```
                              ORG     202H
0202                  OUTBUF  RS      1          ; OUTPUT 1 CHR BUFFER
                      ;
                              ORG     6A4H
06A4                  TEXT_PNT RS     2          ; TEXT TOP ADDRESS
                      ;
                              ORG     6DEH       ; COMPARE ASCII TEXT
06DE                  TXTPNT  RS      2          ;   POINTER
                      ;
                              ORG     1412H
1412                  VRAM    RS      2          ; TEXT VRAM SEGMENT
                      ;
                              ORG     1840H      ; OUTPUT DEVICE FLAG
1840                  OUTFLAG RS      1          ; 4 CRT 3 LPT
                      ;
                              CSEG
                              ORG     0
                      ;
  0007                        BEEP    EQU     7
  0004                        CRT     EQU     4          ; CRT DEVICE NUMBER
                      ;
                      FIND:
0000 3C03                     CMP     AL,3       ; STRING ?
0002 7403        0007         JE      FIND00     ; NO THEN ERROR
0004 E9C500      00CC         JMP     TYPE_MISMATCH_ERROR
                      FIND00:                    ; ELSE
0007 36A1A406                 MOV     AX,WORD PTR TEXT_PNT
000B 36A3DE06                 MOV     WORD PTR TXTPNT,AX
                      ;
000F 16                       PUSH    SS         ; SET TEXT SEGMENT
0010 1F                       POP     DS
0011 C51F                     LDS     BX,[BX] ; STRING DESCREPTER POINTER
                      ;
0013 8A0F                     MOV     CL,[BX] ; GET LENGTH OF STRING
0015 43                       INC     BX
0016 8A07                     MOV     AL,[BX] ; GET RELOCATION CODE
0018 43                       INC     BX
0019 8B37                     MOV     SI,[BX] ; GET STRING OFFSET
001B 0AC0                     OR      AL,AL
001D 7504        0023         JNE     FIND10
001F 8EDA                     MOV     DS,DX
0021 EB02        0025         JMPS    FIND20
                      FIND10:
0023 16                       PUSH    SS
0024 1F                       POP     DS
                      FIND20:
0025 E85C00      0084         CALL    CONV_TEXT ; CONVERT TEXT TO ASCII
                      ;
0028 16                       PUSH    SS
0029 07                       POP     ES
002A BF0302                   MOV     DI,OFFSET TXTBUF
002D E87800      00A8         CALL    COMPARE   ; COMPARE THE STRINGS
                      ;
0030 7303        0035         JNB     FIND40
0032 E80D00      0042         CALL    HIT
                      FIND40:
0035 E83000      0068         CALL    SET_TXTPNT
0038 7205        003F         JB      FIND_END
                      ;
003A E86B01      01A8         CALL    CHECK_STOP
003D EBE6        0025         JMPS    FIND20
                      ;
                      FIND_END:
```

```
003F 33C0                      XOR     AX,AX
0041 CF                        IRET
                      ;
                      HIT:
0042 51                        PUSH    CX
0043 56                        PUSH    SI
0044 1E                        PUSH    DS
                      ;
0045 16                        PUSH    SS
0046 1F                        POP     DS
0047 BE0302                    MOV     SI,OFFSET TXTBUF
                      HIT10:
004A AC                        LODSB
004B 0AC0                      OR      AL,AL
004D 7405          0054        JE      HIT_END
004F E88500        00D7        CALL    OUTCHAR ; OUTPUT CHARACTER
0052 EBF6          004A        JMPS    HIT10
                      HIT_END:
0054 B80D0A                    MOV     AX,0A0DH
0057 36A30202                  MOV     WORD PTR OUTBUF,AX
005B B90200                    MOV     CX,2
005E BF2500                    MOV     DI,37
0061 E83901        019D        CALL    ROM
                      ;
0064 1F                        POP     DS
0065 5E                        POP     SI
0066 59                        POP     CX
0067 C3                        RET
                      ;
                      SET_TXTPNT:
0068 1E                        PUSH    DS
                               ;
0069 16                        PUSH    SS
006A 1F                        POP     DS
006B 368B1EDE06                MOV     BX,WORD PTR TXTPNT
0070 8B07                      MOV     AX,[BX] ; GET LINK POINTER
0072 03D8                      ADD     BX,AX
0074 833F00                    CMP     WORD PTR [BX],0    ; NEXT LINK IS END ?
0077 7408          0081        JE      LINK_END
0079 36891EDE06                MOV     WORD PTR TXTPNT,BX
007E F8                        CLC     ; CF=0
007F EB01          0082        JMPS    LINK_END10
                      LINK_END:
0081 F9                        STC     ; CF=1
                      LINK_END10:
                      ;
0082 1F                        POP     DS
0083 C3                        RET
                      ;
                      ; CONVERT TEXT TO ASCII STRINGS
                      ;
                      ;      INPUT  : TXTPNT [TEXT POINTER]
                      ;     OUTPUT  : BUFFER [0060H:0000H]
                      ;
                      CONV_TEXT:
0084 56                        PUSH    SI      ; SAVE OFFSET STRING
0085 E81000        0098        CALL    CLEAR_BUFFER
0088 BB0302                    MOV     BX,OFFSET TXTBUF ; ASCII TEXT BUF
008B 368B36DE06                MOV     SI,WORD PTR TXTPNT ; TEXT POINTER
0090 BF1000                    MOV     DI,16   ; CONVERT TEXT TO ASCII
0093 E80701        019D        CALL    ROM
0096 5E                        POP     SI      ; LOAD OFFSET STRING
0097 C3                        RET
```

```
                       ;
                       ;   NULL CLEAR OF BUFFER
                       ;
                       CLEAR_BUFFER:
0098 51                    PUSH    CX
                       ;
0099 16                    PUSH    SS
009A 07                    POP     ES
009B BF0302                MOV     DI,OFFSET TXTBUF
009E 33C0                  XOR     AX,AX
00A0 B98000                MOV     CX,128
00A3 FC                    CLD
00A4 F3AB                  REP     STOSW
                       ;
00A6 59                    POP     CX
00A7 C3                    RET
                       ;
                       ;  COMPARE STRINGS WITH ASCII TEXT
                       ;
                       ;    INPUT   : CL [LENGTH OF STRING]
                       ;
                       ;    SOURCE : STRINGS    IN DS:SI
                       ;    DEST.  : ASCII TEXT IN ES:DI
                       ;
                       ;    OUTPUT : CF=1 THEN HIT !!!
                       ;             CF=0 THEN UNMATCH.
                       COMPARE:
00A8 56                    PUSH    SI
00A9 51                    PUSH    CX
                       COMP05:
00AA 57                    PUSH    DI
00AB 56                    PUSH    SI
00AC 51                    PUSH    CX
                       COMP10:
00AD A6                    CMPSB
00AE 7506       00B6       JNE     COMP20    ; DIFFERENT
00B0 FEC9                  DEC     CL
00B2 75F9       00AD       JNE     COMP10
00B4 EB0F       00C5       JMPS    COMP30    ; MATCH
                       COMP20:
00B6 59                    POP     CX
00B7 5E                    POP     SI
00B8 5F                    POP     DI
00B9 47                    INC     DI
00BA 268A05                MOV     AL,ES:[DI] ; END OF LINE ?
00BD 0AC0                  OR      AL,AL
00BF 75E9       00AA       JNE     COMP05
                       ;
00C1 59                    POP     CX
00C2 5E                    POP     SI
00C3 F8                    CLC             ; CF=0 UNMATCH
00C4 C3                    RET
                       COMP30:
00C5 59                    POP     CX
00C6 5E                    POP     SI
00C7 5F                    POP     DI
                       ;
00C8 59                    POP     CX
00C9 5E                    POP     SI
00CA F9                    STC             ; CF=1 MATCH
00CB C3                    RET
                       ;
                       ;  ERROR DISPLAY
                       ;
```

```
                        TYPE_MISMATCH_ERROR:
00CC BF0300                         MOV   DI,3
00CF E8CB00        019D             CALL  ROM
00D2 CF                             IRET
                        ;
                        ;  OUTPUT CHARACTER
                        ;
                        ;    INPUT : AL
                        ;
00D3 00                 F1B      DB      00H       ; ESC FLAG
00D4 00                 KNJ_FLG DB       00H       ; KI KO FLAG
00D5 00                 OUTCHR_NUM DB    00H
00D6 00                 OUTCHR_BUF DB    00H
                        ;
                        OUTCHAR:
00D7 3C1B                           CMP AL,1BH         ; ESC ?
00D9 7507          00E2             JNE OUTCHAR_10     ; YES
00DB 2EC606D30001                   MOV F1B,1          ; ESCAPE FLAG ON
00E1 C3                             RET                ; END
                        OUTCHAR_10:                    ; ELSE
00E2 2E803ED30001                   CMP F1B,1          ; ESC  ON ?
00E8 7523          010D             JNE OUTCHAR_20
00EA B400                           MOV AH,0
00EC 3C4B                           CMP AL,4BH         ; KI ?
00EE 7449          0139             JE OUTCHAR_40      ; YES THEN KNJ_FLG ON.
00F0 2E8826D400                     MOV KNJ_FLG,AH     ; ELSE KO
00F5 2E8826D300                     MOV F1B,AH         ; CLEAR ESC FLAG
00FA 36803E401804                   CMP OUTFLAG,CRT    ; IF NOT CRT THEN
0100 740A          010C             JE OUTCHAR_15      ;  OUTPUT KO CODE
0102 B01B                           MOV AL,1BH
0104 E82200        0129             CALL OUTCHAR_30
0107 B048                           MOV AL,48H
0109 E81D00        0129             CALL OUTCHAR_30
                        OUTCHAR_15:
010C C3                             RET                ; END.
                        OUTCHAR_20:
010D 2E803ED40001                   CMP KNJ_FLG,1      ; KI ?
0113 7514          0129             JNE OUTCHAR_30
0115 2E803ED50000                   CMP OUTCHR_NUM,0  ; FIRST CHAR ?
011B 753C          0159             JNE OUTCHAR_60     ; NO THEN OUTPUT 2 BYTES
011D 2C20                           SUB AL,20H
011F 2EA2D600                       MOV OUTCHR_BUF,AL;ELSE SAVE FIRST CHAR
0123 2EFE06D500                     INC OUTCHR_NUM     ; COUNTER+1
0128 C3                             RET
                        ;
                        OUTCHAR_30:                        ; NORMAL OUTPUT CHR
0129 51                             PUSH      CX
012A B90100                         MOV       CX,1         ; ONE CHARACTER
012D 36A20202                       MOV       OUTBUF,AL    ; SET BUFFER CHR.
0131 BF2500                         MOV       DI,37        ; OUTPUT CURRENT DEVICE.
0134 E86600        019D             CALL      ROM
0137 59                             POP       CX
0138 C3                             RET
                        ;
                        OUTCHAR_40:
0139 FEC4                           INC AH
013B 2E8826D400                     MOV KNJ_FLG,AH     ; KI ON
0140 36803E401804                   CMP OUTFLAG,CRT    ; OUTPUT DEVICE IS CRT ?
0146 740A          0152             JE OUTCHAR_50      ; YES THEN END.
0148 B01B                           MOV AL,1BH
014A E8DCFF        0129             CALL OUTCHAR_30
014D B04B                           MOV AL,4BH
014F E8D7FF        0129             CALL OUTCHAR_30
```

```
                        OUTCHAR_50:
0152 2EC606D30000              MOV F1B,0        ; CLEAR ESC FLAG
0158 C3                        RET
                        OUTCHAR_60:
0159 8AE0                      MOV AH,AL
015B 2EA0D600                  MOV AL,OUTCHR_BUF
015F E80700        0169        CALL OUTCHAR_II
0162 2EC606D50000              MOV OUTCHR_NUM,0          ; CLEAR COUNTER
0168 C3                        RET
                        ;
                        ;    OUTCHAR PART II
                        ;
                        OUTCHAR_II:
0169 51                        PUSH CX
016A 1E                        PUSH DS
016B 06                        PUSH ES
016C 16                        PUSH SS
016D 1F                        POP  DS
016E 36803E401804              CMP  OUTFLAG,CRT
0174 7516          018C        JNE  OUTCHAR_II_10
                        ;
0176 36A30000                  MOV WORD PTR PRNTBUF,AX
017A 36A30200                  MOV WORD PTR PRNTBUF+2,AX
017E 368E061214                MOV ES,WORD PTR VRAM
0183 B90400                    MOV CX,4
0186 CD89                      INT 89H
                        OUTCHAR_II_05:
0188 07                        POP ES
0189 1F                        POP DS
018A 59                        POP CX
018B C3                        RET
                        OUTCHAR_II_10:
018C 0420                      ADD AL,20H
018E 36A30202                  MOV WORD PTR OUTBUF,AX
0192 BF2500                    MOV DI,37
0195 B90200                    MOV CX,2
0198 E80200        019D        CALL ROM
019B EBEB          0188        JMPS OUTCHAR_II_05
                        ;
                        ; ROM CALL
                        ;
                        ROM:
019D 1E                        PUSH    DS       ; SAVE RESISTERS
019E 56                        PUSH    SI
019F 51                        PUSH    CX
                        ;
01A0 16                        PUSH    SS       ; CALL IN ROM ROUTINE.
01A1 1F                        POP     DS
01A2 CDC4                      INT     0C4H
                        ;
01A4 59                        POP     CX
01A5 5E                        POP     SI
01A6 1F                        POP     DS
01A7 C3                        RET
                        ;
                        ;  CHECK STOP KEY
                        ;
                        ;      IF STOP KEY ON THEN CF=1
                        ;                          ELSE CF=0
                        CHECK_STOP:
01A8 BF4B00                    MOV     DI,75
01AB E8EFFF        019D        CALL    ROM
01AE C3                        RET
```

```
;
            END
```

15-2 リプレイス(文字列の置き換え)

PRINTをLPRINTに変えたり，サブルーチンで同じ変数名を使ってしまった場合など，変数名を変えたりなどよく出くわす事態です。こんなときこのコマンドが役立ちます。

リプレイスまたはチェンジコマンドというのは，テキスト内の文字列1を文字列2で置きかえるコマンドのことです。

使い方は，

CMD　〝文字列1″，〝文字列2″

とすると，RAM上のBASICテキスト内の文字列1を文字列2でおきかえます。もし，128バイトを超える場合は，

String too long

と表示して停止し，ダイレクトモードに戻ります。文字列は必らず，ダブルクォテーション（″）で囲んで下さい。文字列2をヌル・ストリング（〝″）にすると，文字列1を削除します。

CMD　〝文字列1″

とすると，文字列1のサーチとなり，テキストは書き換えません。テキストサーチと同じになります。まず，これで確認したあと，リプレイスするとよいでしょう。

さて，プログラムですが，テキストサーチと同様にして文字列1を見つけたあと，文字列1を削除（DEL）し，そこへ文字列2を挿入（INS）し，コマンド77（4DH）〝1行トランスレート″でRAM上のBASICテキストに出力する（TRANS）という方法をとっています。

このコマンド77（4DH）を使うと，他に，DATA文の自動作成等もマシン語で書けます。

明らさまに，ストップキーチェックは行っていませんが，コマンド76(4CH)のキーセンスさえすれば，CTRL -Sや STOP キー処理をしてくれます。

詳しくはソースリストの方を見て下さい。内部ルーチンの利用等，参考になると思います。

ディスクコードは，セグメント1000HのRAM上にありますが，ROM内ルーチンを使っているのです。皆さんも，ROM内ルーチンを有効に使って下さい。

リプレイスプログラム

```
0 'SAVE"REPLACE"
100 '*****************************************************
110 '*                                                   *
120 '*     REPLACE STRINGS TO ANOTHER ONE .              *
130 '*                                                   *
140 '*        USAGE IS FOLLOWING ...                     *
150 '*                                                   *
160 '*           CMD "STRING1","STRING2"                 *
170 '*                                                   *
180 '*             REPLACE STRING1 -> STRING2 .          *
190 '*                                                   *
200 '*  Copyright All Reserved by SYSTEM-SOFT(C)         *
210 '*                                                   *
220 '*****************************************************
230 PRINT "Now make 'CMD' function, just wait ..."
240  DEF SEG=&H60
250   POKE &H1593,0 : ' CLEAR EXPANDED COMMAND FLAG.
260   POKE &H159C,0:POKE &H159D,0 : ' SET CMD REPLACE ADDRESS
270   POKE &H159E,0:POKE &H159F,&H1F
280   POKE &H15A0,&HF:POKE &H15A1,4 ' SET IEEE RETF
290   POKE &H15A2,0:POKE &H15A3,&H1F
300  CLEAR 300,&H1F00:DEF SEG=&H1F00
310   FOR I=0 TO &H40F
320    READ A$:POKE I,VAL("&H"+A$)
330   NEXT
340  DEF SEG=&H60
350  POKE &H1593,1 : ' SET EXPANDED FLAG.
360 A$=CHR$(&H22)
370 PRINT "Now you use ;"
380 PRINT "   'CMD "A$"STRING1"A$","A$"STRING2"A$" ' command. "
390 BEEP 1:A=LOG(10):BEEP 0
400 END
10000 DATA 3C,E8,75,05,E9,08,01,00,00,F8,CB,00,00,00,00,00:'0000H
10010 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00:'0010H
10020 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00:'0020H
10030 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00:'0030H
10040 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00:'0040H
10050 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00:'0050H
10060 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00:'0060H
10070 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00:'0070H
10080 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00:'0080H
10090 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00:'0090H
10100 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00:'00A0H
10110 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00:'00B0H
10120 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00:'00C0H
10130 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00:'00D0H
10140 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00:'00E0H
10150 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00:'00F0H
10160 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,2E:'0100H
10170 DATA C6,06,0B,00,00,2E,C6,06,8C,00,00,46,E8,55,00,3C:'0110H
10180 DATA 22,75,69,BF,0C,00,0E,07,33,C9,46,AC,3C,00,74,67:'0120H
10190 DATA 3C,22,74,0F,FE,C1,80,F9,81,74,48,2E,88,0E,0B,00:'0130H
10200 DATA AA,EB,E8,E8,2E,00,3C,2C,75,49,46,E8,26,00,3C,00:'0140H
10210 DATA 74,45,3C,22,75,36,33,C9,BF,8D,00,46,AC,3C,00,74:'0150H
10220 DATA 36,3C,22,74,32,FE,C1,80,F9,81,74,17,2E,88,0E,8C:'0160H
10230 DATA 00,AA,EB,E8,AC,3C,00,74,08,3C,0B,72,F7,3C,20,74:'0170H
10240 DATA F3,4E,C3,B0,0F,BF,00,00,E8,D2,01,C3,BF,02,00,E8:'0180H
10250 DATA CB,01,C3,B0,01,EB,02,32,C0,2E,A2,0E,01,E8,D4,FF:'0190H
10260 DATA 36,A1,A4,06,2E,A3,07,00,BE,0C,00,2E,8A,0E,0B,00:'01A0H
10270 DATA 0E,1F,E8,92,00,2E,C6,06,0D,01,00,16,07,BF,03,02:'01B0H
10280 DATA E8,42,02,E8,A5,00,73,16,2E,C6,06,0D,01,01,2E,F6:'01C0H
```

```
10290 DATA 06,0E,01,01,75,08,E8,96,01,E8,B5,01,EB,E5,2E,80:'01D0H
10300 DATA 3E,0D,01,01,75,0E,2E,F6,06,0E,01,01,75,03,E8,ED:'01E0H
10310 DATA 01,E8,11,00,E8,34,00,72,05,E8,6C,01,EB,B4,BF,1E:'01F0H
10320 DATA 00,E8,59,01,CB,51,56,1E,16,1F,BE,03,02,AC,0A,C0:'0200H
10330 DATA 74,05,E8,85,00,EB,F6,B8,0D,0A,36,A3,02,02,B9,02:'0210H
10340 DATA 00,BF,25,00,E8,36,01,1F,5E,59,C3,1E,16,1F,2E,8B:'0220H
10350 DATA 1E,07,00,8B,07,03,D8,83,3F,00,74,08,2E,89,1E,07:'0230H
10360 DATA 00,F8,EB,01,F9,1F,C3,56,E8,10,00,BB,03,02,2E,8B:'0240H
10370 DATA 36,07,00,BF,10,00,E8,04,01,5E,C3,51,16,07,BF,03:'0250H
10380 DATA 02,33,C0,B9,80,00,FC,F3,AB,59,C3,56,51,57,56,51:'0260H
10390 DATA A6,75,06,FE,C9,75,F9,EB,0F,59,5E,5F,47,26,8A,05:'0270H
10400 DATA 0A,C0,75,E9,59,5E,F8,C3,59,5E,5F,59,5E,F9,C3,BF:'0280H
10410 DATA 03,00,E8,C8,00,CF,00,00,00,00,3C,1B,75,07,2E,C6:'0290H
10420 DATA 06,96,02,01,C3,2E,80,3E,96,02,01,75,23,32,E4,3C:'02A0H
10430 DATA 4B,74,49,2E,88,26,97,02,2E,88,26,96,02,36,80,3E:'02B0H
10440 DATA 40,18,04,74,0A,B0,1B,E8,22,00,B0,48,E8,1D,00,C3:'02C0H
10450 DATA 2E,80,3E,97,02,01,75,14,2E,80,3E,98,02,00,75,3C:'02D0H
10460 DATA 2C,20,2E,A2,99,02,2E,FE,06,98,02,C3,51,B9,01,00:'02E0H
10470 DATA 36,A2,02,02,BF,25,00,E8,63,00,59,C3,FE,C4,2E,88:'02F0H
10480 DATA 26,97,02,36,80,3E,40,18,04,74,0A,B0,1B,E8,DC,FF:'0300H
10490 DATA B0,4B,E8,D7,FF,2E,C6,06,96,02,00,C3,8A,E0,2E,A0:'0310H
10500 DATA 99,02,E8,07,00,2E,C6,06,98,02,00,C3,1E,06,16,1F:'0320H
10510 DATA 51,36,80,3E,40,18,04,75,16,36,A3,00,00,36,A3,02:'0330H
10520 DATA 00,36,8E,06,12,14,B9,04,00,CD,89,59,07,1F,C3,50:'0340H
10530 DATA 04,20,E8,97,FF,58,8A,C4,E8,91,FF,EB,EE,1E,56,51:'0350H
10540 DATA 16,1F,CD,C4,59,5E,1F,C3,BF,4B,00,E8,EF,FF,C3,33:'0360H
10550 DATA C9,2E,8A,0E,0B,00,83,F9,00,75,01,C3,1E,56,06,1F:'0370H
10560 DATA 8B,D7,8B,F7,03,F1,AC,AA,3C,00,75,FA,5E,1F,8B,FA:'0380H
10570 DATA C3,33,C9,2E,8A,0E,8C,00,83,F9,00,75,06,2E,8A,0E:'0390H
10580 DATA 0B,00,C3,1E,56,06,1F,8B,F7,8B,D9,33,C9,8B,D7,AC:'03A0H
10590 DATA 41,3C,00,75,FA,4E,8B,FE,03,FB,81,FF,01,03,72,03:'03B0H
10600 DATA E9,C0,FD,FD,F3,A4,FC,8B,CB,06,0E,1F,16,07,8B,FA:'03C0H
10610 DATA BE,8D,00,F3,A4,07,5E,1F,2E,8A,0E,0B,00,C3,1E,51:'03D0H
10620 DATA 56,16,1F,BE,02,02,C6,04,20,46,33,C9,AC,41,3C,00:'03E0H
10630 DATA 75,FA,BF,4D,00,E8,65,FF,5E,59,1F,C3,3C,30,72,03:'03F0H
10640 DATA 3C,3A,C3,F5,C3,26,8A,05,47,E8,F0,FF,72,F7,C3,CB:'0400H
```

マシン語のソースリストは次のとおりです。

```
                         ;
                         ;    THIS COMMAND CHANGES STRINGS
                         ;
                         ;    CALLING SEQUENCE
                         ;
                         ;       CMD 'STRING1','STRING2'
                         ;
 00E8                            CMD     EQU      0E8H
                         ;
                                 SSEG
                                 ORG      0000H
0000                     PRNTBUF RS       512      ; PRINT BUFFER
                         ;
                                 ORG      203H
0203                     TXTBUF  RS       256      ; ASCII TEXT BUFFER
                         ;
                                 ORG      202H
0202                     OUTBUF  RS       1        ; OUTPUT 1 CHR BUFFER
                         ;
                                 ORG      6A4H
06A4                     TEXT_PNT RS      2        ; TEXT TOP ADDRESS
                         ;
```

```
                         ORG     6E8H
06E8          EXEC_TXT_COPY      RS      2
              ;
                         ORG     6EAH
06EA          EXEC_TXT           RS      2
              ;
                         ORG     1406H
1406          TEXT_LINE          RS      2 ; TRANSLATE BUFFER POINTER
              ;
                         ORG     1412H
1412          VRAM       RS      2       ; TEXT VRAM SEGMENT
              ;
                         ORG     1840H   ; OUTPUT DEVICE FLAG
1840          OUTFLAG RS         1       ; 4 CRT 3 LPT
              ;
                         CSEG
                         ORG     0
              ;
 0004                    CRT     EQU     4       ; CRT DEVICE NUMBER
              ;
              CHG:
0000 3CE8                CMP     AL,CMD
0002 7505        0009    JNE     CHG05
0004 E90801      010F    JMP     CHG10
0007 0000     TXTPNT     DW      0000H   ; COMPARE ASCII TEXT
                                         ; POINER
              ;
              CHG05:
0009 F8                  CLC
000A CB                  RETF
              ;
000B 00       BUF1       DB      00H     ; LEN(STR1)
000C                     RS      128     ; STRING BUFFER 1
008C 00       BUF2       DB      00H     ; LEN(STR2)
008D                     RS      128     ; STRING BUFFER 2
              ;
010D 00       HIT_FLAG   DB      00H     ; ON WHEN HIT THE STR.
010E 00       NO_CHG_FLG DB      00H     ; FIND ONLY FLAG.
              ;
              CHG10:
010F 2EC6060B0000        MOV     BUF1,0  ; CLEAR BUFFERS
0115 2EC6068C0000        MOV     BUF2,0
              ;
011B 46                  INC     SI
011C E85500      0174    CALL    SKIP_SPC
011F 3C22                CMP     AL,'"'  ; FIRST CHR IS " ?
0121 7569        018C    JNE     ILEGAL  ; NO THEN ERROR.
0123 BF0C00              MOV     DI,OFFSET BUF1+1
0126 0E                  PUSH    CS
0127 07                  POP     ES
0128 33C9                XOR     CX,CX   ; CLEAR COUNTER
012A 46                  INC     SI
              CHG20:
012B AC                  LODSB
012C 3C00                CMP     AL,00H
012E 7467        0197    JE      GO_MAIN
0130 3C22                CMP     AL,'"'
0132 740F        0143    JE      CHG30
0134 FEC1                INC     CL
0136 80F981              CMP     CL,129
0139 7448        0183    JE      OVER
013B 2E880E0B00          MOV     BUF1,CL
0140 AA                  STOSB
0141 EBE8        012B    JMPS    CHG20
```

```
                        CHG30:
0143 E82E00     0174           CALL    SKIP_SPC
0146 3C2C                      CMP     AL,','
0148 7549       0193           JNE     NO_CHG   ; CMD STRING IS NO CHANGE
014A 46                        INC     SI
014B E82600     0174           CALL    SKIP_SPC
014E 3C00                      CMP     AL,00H
0150 7445       0197           JE      GO_MAIN
0152 3C22                      CMP     AL,'"'
0154 7536       018C           JNE     ILEGAL
0156 33C9                      XOR     CX,CX    ; CLEAR COUNTER
0158 BF8D00                    MOV     DI,OFFSET BUF2+1
015B 46                        INC     SI
                        CHG40:
015C AC                        LODSB
015D 3C00                      CMP     AL,00H
015F 7436       0197           JE      GO_MAIN
0161 3C22                      CMP     AL,'"'
0163 7432       0197           JE      GO_MAIN
0165 FEC1                      INC     CL
0167 80F981                    CMP     CL,129
016A 7417       0183           JE      OVER
016C 2E880E8C00                MOV     BUF2,CL
0171 AA                        STOSB
0172 EBE8       015C           JMPS    CHG40
                        ;
                        SKIP_SPC:
0174 AC                        LODSB
0175 3C00                      CMP     AL,00H
0177 7408       0181           JE      SKIP_SPC_END
0179 3C0B                      CMP     AL,0BH
017B 72F7       0174           JB      SKIP_SPC
017D 3C20                      CMP     AL,' '
017F 74F3       0174           JE      SKIP_SPC
                        SKIP_SPC_END:
0181 4E                        DEC     SI
0182 C3                        RET
                        ;
                        OVER:
0183 B00F                      MOV     AL,15    ; STRING TOO LONG
0185 BF0000                    MOV     DI,0     ; DISPLAY ERROR MES.
0188 E8D201     035D           CALL    ROM
018B C3                        RET
                        ;
                        ILEGAL:
018C BF0200                    MOV     DI,2     ; ILLEGAL FUNC. CALL
018F E8CB01     035D           CALL    ROM
0192 C3                        RET
                        ;
                        NO_CHG:
0193 B001                      MOV     AL,1     ; ONLY FIND
0195 EB02       0199           JMPS    GO_MAIN10
                        GO_MAIN:
0197 32C0                      XOR     AL,AL    ; EXCHANGE
                        GO_MAIN10:
0199 2EA20E01                  MOV     NO_CHG_FLG,AL
019D E8D4FF     0174           CALL    SKIP_SPC
                        ;
                        ;     MAIN PROGRAM
                        ;
01A0 36A1A406                  MOV     AX,WORD PTR TEXT_PNT
01A4 2EA30700                  MOV     WORD PTR TXTPNT,AX
01A8 BE0C00                    MOV     SI,OFFSET BUF1+1 ; STRING1 BUF
```

```
01AB 2E8A0E0B00              MOV     CL,BUF1 ; LENGTH OF STRING1
01B0 0E                      PUSH    CS      ; SEGMENT OF STRING BUF
01B1 1F                      POP     DS
                     FIND20:
01B2 E89200     0247         CALL    CONV_TEXT ; CONVERT TEXT TO ASCII
                     ;
01B5 2EC6060D0100            MOV     HIT_FLAG,0
01BB 16                      PUSH    SS
01BC 07                      POP     ES
01BD BF0302                  MOV     DI,OFFSET TXTBUF
01C0 E84202     0405         CALL    SKIP_LINE_NO
                     FIND30:                         ; REPEAT
01C3 E8A500     026B         CALL    COMPARE         ;   COMPARE THE STRINGS
                     ;
01C6 7316       01DE         JNB     FIND40          ;
01C8 2EC6060D0101            MOV     HIT_FLAG,1      ;   FIND THE STRING !
01CE 2EF6060E0101            TEST    NO_CHG_FLG,1    ;   IF NO CHANGE THEN
01D4 7508       01DE         JNE     FIND40          ;    SKIP DELETE & INSERT.
01D6 E89601     036F         CALL    DEL             ;   ELSE
01D9 E8B501     0391         CALL    INS             ;
01DC EBE5       01C3         JMPS    FIND30          ;   ENDIF
                     FIND40:
01DE 2E803E0D0101            CMP     HIT_FLAG,1      ;   IF NOT FOUND THEN
01E4 750E       01F4         JNE     FIND50          ;     NEXT
01E6 2EF6060E0101            TEST    NO_CHG_FLG,1    ;   ELSEIF NO CHANGE THEN
01EC 7503       01F1         JNE     FIND45          ;    SKIP TRANSLATE TO TXT
01EE E8ED01     03DE         CALL    TRANS           ;     ELSE TRANSLATE
                     FIND45:                         ;     ENDIF
                                                     ;      DISPLAY LINE.
01F1 E81100     0205         CALL    HIT             ;   ENDIF
                     FIND50:
01F4 E83400     022B         CALL    SET_TXTPNT      ; UNTIL
                                                     ;   (DETECT END OF TEXT)
01F7 7205       01FE         JB      FIND_END
                     ;
01F9 E86C01     0368         CALL    CHECK_STOP
01FC EBB4       01B2         JMPS    FIND20
                     ;
                     FIND_END:
01FE BF1E00                  MOV     DI,30    ; DIRECT MODE ENTRY !!
0201 E85901     035D         CALL    ROM
0204 CB                      RETF    ; FOR FALE SAFE.
                     ;
                     ;       DISPLAY THE LINE
                     ;
                     HIT:
0205 51                      PUSH    CX       ; SAVE RESISTERS
0206 56                      PUSH    SI
0207 1E                      PUSH    DS
                     ;
0208 16                      PUSH    SS       ; SET [DS]
0209 1F                      POP     DS
020A BE0302                  MOV     SI,OFFSET TXTBUF
                     HIT10:
020D AC                      LODSB            ; WHILE CHR<>NULL DO
020E 0AC0                    OR      AL,AL    ;
C210 7405       0217         JE      HIT_END  ;
0212 E88500     029A         CALL    OUTCHAR  ;   OUTPUT CHARACTER
0215 EBF6       020D         JMPS    HIT10    ; ENDWHILE
                     HIT_END:
0217 B80D0A                  MOV     AX,0A0DH ; OUTPUT DELIMITER.
```

```
021A 36A30202                  MOV     WORD PTR OUTBUF,AX
021E B90200                    MOV     CX,2     ; 2 CHRS
0221 BF2500                    MOV     DI,37    ; OUTPUT CHRS
0224 E83601        035D        CALL    ROM
                        ;
0227 1F                        POP     DS       ; LOAD RESISTERS
0228 5E                        POP     SI
0229 59                        POP     CX
022A C3                        RET
                        ;
                        SET_TXTPNT:
022B 1E                        PUSH    DS       ; SAVE [DS]
                               ;
022C 16                        PUSH    SS       ; SET [DS]
022D 1F                        POP     DS
022E 2E8B1E0700                MOV     BX,WORD PTR TXTPNT
0233 8B07                      MOV     AX,[BX] ; GET LINK POINTER
0235 03D8                      ADD     BX,AX
0237 833F00                    CMP     WORD PTR [BX],0 ; NEXT LINK IS END ?
023A 7408          0244        JE      LINK_END        ; IF NO THEN
023C 2E891E0700                MOV     WORD PTR TXTPNT,BX
0241 F8                        CLC                     ;   CF=0
0242 EB01          0245        JMPS    LINK_END10      ; ELSE
                        LINK_END:                      ;
0244 F9                        STC                     ;   CF=1
                        LINK_END10:                    ; ENDIF
                               ;
0245 1F                        POP     DS       ; LOAD [DS]
0246 C3                        RET
                        ;
                        ; CONVERT TEXT TO ASCII STRINGS
                        ;
                        ;     INPUT : TXTPNT [TEXT POINTER]
                        ;    OUTPUT : BUFFER [0060H:0203H]
                        ;
                        CONV_TEXT:
0247 56                        PUSH    SI              ; SAVE OFFSET
                                                       ;  OF THE STRING POINTER.
0248 E81000        025B        CALL    CLEAR_BUFFER
024B BB0302                    MOV     BX,OFFSET TXTBUF  ; ASCII TEXT LINE BUF
024E 2E8B360700                MOV     SI,WORD PTR TXTPNT ; TEXT LINE POINTER
0253 BF1000                    MOV     DI,16           ; CONVERT TEXT TO ASCII
0256 E80401        035D        CALL    ROM
0259 5E                        POP     SI              ; LOAD OFFSET
025A C3                        RET                     ;  OF THE STRING POINTER.
                        ;
                        ;   NULL CLEAR OF BUFFER
                        ;
                        CLEAR_BUFFER:
025B 51                        PUSH    CX
                        ;
025C 16                        PUSH    SS
025D 07                        POP     ES
025E BF0302                    MOV     DI,OFFSET TXTBUF
0261 33C0                      XOR     AX,AX
0263 B98000                    MOV     CX,128   ; FOR CX=128 TO 0 STEP -1
0266 FC                        CLD              ;   WORD PTR [DI]=0000H
0267 F3AB                      REP     STOSW    ;
                                                ; NEXT
0269 59                        POP     CX
026A C3                        RET
                        ;
                        ;  COMPARE STRINGS WITH ASCII TEXT
```

```
                          ;
                          ;     INPUT  : CL [LENGTH OF STRING]
                          ;
                          ;     SOURCE : STRINGS    IN DS:SI
                          ;     DEST.  : ASCII TEXT IN ES:DI
                          ;
                          ;     OUTPUT : CF=1 THEN HIT !!!
                          ;              CF=0 THEN UNMATCH.
                          COMPARE:
026B 56                          PUSH    SI
026C 51                          PUSH    CX
                          COMP05:
026D 57                          PUSH    DI
026E 56                          PUSH    SI
026F 51                          PUSH    CX
                          COMP10:
0270 A6                          CMPSB
0271 7506          0279          JNE     COMP20   ; DIFFERENT
0273 FEC9                        DEC     CL
0275 75F9          0270          JNE     COMP10
0277 EB0F          0288          JMPS    COMP30   ; MATCH
                          COMP20:
0279 59                          POP     CX
027A 5E                          POP     SI
027B 5F                          POP     DI
027C 47                          INC     DI
027D 268A05                      MOV     AL,ES:[DI] ; END OF LINE ?
0280 0AC0                        OR      AL,AL
0282 75E9          026D          JNE     COMP05
                          ;
0284 59                          POP     CX
0285 5E                          POP     SI
0286 F8                          CLC     ; CF=0 UNMATCH
0287 C3                          RET
                          COMP30:
0288 59                          POP     CX
0289 5E                          POP     SI
028A 5F                          POP     DI
                          ;
028B 59                          POP     CX
028C 5E                          POP     SI
028D F9                          STC     ; CF=1 MATCH
028E C3                          RET
                          ;
                          ;  ERROR DISPLAY
                          ;
                          TYPE_MISMATCH_ERROR:
028F BF0300                      MOV     DI,3     ; TYPE MISMATCH ERROR
0292 E8C800        035D          CALL    ROM
0295 CF                          IRET
                          ;
                          ;  OUTPUT CHARACTER
                          ;
                          ;    INPUT : AL
                          ;
0296 00                   F1B    DB      00H      ; ESC FLAG
0297 00                   KNJ_FLG DB     00H      ; KI KO FLAG
0298 00                   OUTCHR_NUM DB  00H
0299 00                   OUTCHR_BUF DB  00H
                          ;
                          OUTCHAR:
029A 3C1B                        CMP AL,1BH       ; ESC ?
029C 7507          02A5          JNE OUTCHAR_10   ; YES
```

```
029E 2EC606960201             MOV F1B,1          ; ESCAPE FLAG ON
02A4 C3                       RET                ; END.
                    OUTCHAR_10:                  ; ELSE
02A5 2E803E960201             CMP F1B,1          ; ESC ON ?
02AB 7523           02D0      JNE OUTCHAR_20
02AD 32E4                     XOR AH,AH
02AF 3C4B                     CMP AL,4BH         ; KI ?
02B1 7449           02FC      JE OUTCHAR_40      ; YES THEN KNJ_FLG ON.
02B3 2E88269702               MOV KNJ_FLG,AH     ; ELSE KO
02B8 2E88269602               MOV F1B,AH         ; CLEAR ESC FLAG
02BD 36803E401804             CMP OUTFLAG,CRT    ; OUTPUT DEVICE IS CRT ?
02C3 740A           02CF      JE OUTCHAR_15      ; YES END.
02C5 B01B                     MOV AL,1BH         ; NO OUTPUT KO CODE
02C7 E82200         02EC      CALL OUTCHAR_30
02CA B048                     MOV AL,48H
02CC E81D00         02EC      CALL OUTCHAR_30
                    OUTCHAR_15:
02CF C3                       RET                ; END.
                    OUTCHAR_20:
02D0 2E803E970201             CMP KNJ_FLG,1      ; KI
02D6 7514           02EC      JNE OUTCHAR_30
02D8 2E803E980200             CMP OUTCHR_NUM,0 ; FIRST CHAR ?
02DE 753C           031C      JNE OUTCHAR_60     ; NO THEN OUTPUT 2 BYTES
02E0 2C20                     SUB AL,20H         ; ELSE
02E2 2EA29902                 MOV OUTCHR_BUF,AL; SAVE FIRST CHAR
02E6 2EFE069802               INC OUTCHR_NUM     ; COUNTER+1
02EB C3                       RET                ; END.
                    ;
                    OUTCHAR_30:                  ; NORMAL OUTPUT
02EC 51                       PUSH    CX
02ED B90100                   MOV     CX,1       ; ONE CHARACTER
02F0 36A20202                 MOV     OUTBUF,AL  ; SET BUFFER CHR.
02F4 BF2500                   MOV     DI,37      ; OUTPUT CURRENT DEVICE.
02F7 E86300         035D      CALL    ROM
02FA 59                       POP     CX
02FB C3                       RET
                    ;
                    OUTCHAR_40:
02FC FEC4                     INC AH
02FE 2E88269702               MOV KNJ_FLG,AH     ; KI
0303 36803E401804             CMP OUTFLAG,CRT    ; OUTPUT DEVICE IS CRT ?
0309 740A           0315      JE OUTCHAR_50      ; YES THEN END.
030B B01B                     MOV AL,1BH         ; OUT TO THE DEVICE KI.
030D E8DCFF         02EC      CALL OUTCHAR_30
0310 B04B                     MOV AL,4BH
0312 E8D7FF         02EC      CALL OUTCHAR_30
                    OUTCHAR_50:
0315 2EC606960200             MOV F1B,0          ; CLEAR ESC FLAG
031B C3                       RET                ; END.
                    OUTCHAR_60:
031C 8AE0                     MOV AH,AL
031E 2EA09902                 MOV AL,OUTCHR_BUF
0322 E80700         032C      CALL OUTCHAR_II
0325 2EC606980200             MOV OUTCHR_NUM,0          ; CLEAR COUNTER
032B C3                       RET                       ; END.
                    ;
                    ;    OUTCHAR PART II [KANJI]
                    ;
                    OUTCHAR_II:
032C 1E                       PUSH DS
032D 06                       PUSH ES
032E 16                       PUSH SS
032F 1F                       POP  DS
```

```
0330 51                          PUSH CX
0331 36803E401804                CMP OUTFLAG,CRT
0337 7516          034F          JNE OUTCHAR_II_10
                           ;
0339 36A30000                    MOV WORD PTR PRNTBUF,AX
033D 36A30200                    MOV WORD PTR PRNTBUF+2,AX
0341 368E061214                  MOV ES,WORD PTR VRAM
0346 B90400                      MOV CX,4
0349 CD89                        INT 89H
                           OUTCHAR_II_05:
034B 59                          POP CX
034C 07                          POP ES
034D 1F                          POP DS
034E C3                          RET
                           OUTCHAR_II_10:
034F 50                          PUSH AX
0350 0420                        ADD AL,20H
0352 E897FF        02EC          CALL OUTCHAR_30
0355 58                          POP AX
0356 8AC4                        MOV AL,AH
0358 E891FF        02EC          CALL OUTCHAR_30
035B EBEE          034B          JMPS OUTCHAR_II_05
                           ;
                           ;  INTERPRITER ROM CALL
                           ;
                           ROM:
035D 1E                          PUSH    DS       ; SAVE RESISTERS
035E 56                          PUSH    SI
035F 51                          PUSH    CX
                           ;
0360 16                          PUSH    SS       ; CALL IN ROM ROUTINE.
0361 1F                          POP     DS
0362 CDC4                        INT     0C4H
                           ;
0364 59                          POP     CX
0365 5E                          POP     SI
0366 1F                          POP     DS
0367 C3                          RET
                           ;
                           ; CHECK STOP KEY
                           ;
                           CHECK_STOP:
0368 BF4B00                      MOV     DI,75    ; KEY SENSE
036B E8EFFF        035D          CALL    ROM
036E C3                          RET
                           ;
                           ; DELETE STRING1 IN TEXT
                           ;
                           DEL:
036F 33C9                        XOR     CX,CX
0371 2E8A0E0B00                  MOV     CL,BUF1  ; IF LENGTH IS ZERO THEN
0376 83F900                      CMP     CX,0000H ;   END.
0379 7501          037C          JNE     DEL10
037B C3                          RET
                           DEL10:
037C 1E                          PUSH    DS       ; SAVE RESISTERS
037D 56                          PUSH    SI
                           ;
037E 06                          PUSH    ES
037F 1F                          POP     DS
0380 8BD7                        MOV     DX,DI    ; SAVE FIRST POINTER IN [DX]
0382 8BF7                        MOV     SI,DI    ; END POINT OF
0384 03F1                        ADD     SI,CX    ;   THE STRING IN TEXT.
```

```
                    DEL20:
0386 AC                     LODSB              ; REPEAT
0387 AA                     STOSB              ;   MOVE CHR.
0388 3C00                   CMP     AL,00H     ; UNTIL CHR=' '
038A 75FA          0386     JNE     DEL20
                    ;
038C 5E                     POP     SI         ; LOAD RESISTERS
038D 1F                     POP     DS
038E 8BFA                   MOV     DI,DX      ; SET POINTER .
0390 C3                     RET
                    ;
                    ;  INSERT STRING2 IN TEXT
                    ;
                    INS:
0391 33C9                   XOR     CX,CX
0393 2E8A0E8C00             MOV     CL,BUF2 ; IF LENGTH IS ZERO THEN
0398 83F900                 CMP     CX,0000H ;   END.
039B 7506          03A3     JNE     INS10
039D 2E8A0E0B00             MOV     CL,BUF1
03A2 C3                     RET
                    INS10:
03A3 1E                     PUSH    DS         ; SAVE RESISTERS PART-1
03A4 56                     PUSH    SI         ; SOURCE STRING2
                    ;
03A5 06                     PUSH    ES         ; SET DS=ES(=60H)
03A6 1F                     POP     DS
03A7 8BF7                   MOV     SI,DI
03A9 8BD9                   MOV     BX,CX      ; SAVE LEN(STR2$)
03AB 33C9                   XOR     CX,CX
03AD 8BD7                   MOV     DX,DI
                    INS20:
03AF AC                     LODSB
03B0 41                     INC     CX         ; COUNT REMAINDER
03B1 3C00                   CMP     AL,00H
03B3 75FA          03AF     JNE     INS20
03B5 4E                     DEC     SI
03B6 8BFE                   MOV     DI,SI
03B8 03FB                   ADD     DI,BX
03BA 81FF0103               CMP     DI,301H
03BE 7203          03C3     JB      INS30
03C0 E9C0FD        0183     JMP     OVER
                    INS30:
03C3 FD                     STD                ; DECREASE MOVEMENT
03C4 F3A4                   REP     MOVSB      ; MAKE SPACES FOR STR2$
03C6 FC                     CLD                ; SET NORMAL INCREASE
03C7 8BCB                   MOV     CX,BX      ; LOAD LEN(STR2$)
                    ;
03C9 06                     PUSH    ES         ; SAVE RESISTER PART-2
                    ;
03CA 0E                     PUSH    CS         ; INSERT THE STRING2
03CB 1F                     POP     DS
03CC 16                     PUSH    SS
03CD 07                     POP     ES
03CE 8BFA                   MOV     DI,DX      ; POINT OF INSERT PART
03D0 BE8D00                 MOV     SI,OFFSET BUF2+1 ; STRING2 TOP
03D3 F3A4                   REP     MOVSB      ; GO INSERT !
                    ;
03D5 07                     POP     ES         ; LOAD RESISTER PART-2
                    ;
03D6 5E                     POP     SI         ; LOAD RESISTERS PART-1
03D7 1F                     POP     DS
03D8 2E8A0E0B00             MOV     CL,BUF1    ; CL=LEN(STR1$)
03DD C3                     RET
```

```
                          ;
                          ;   TRANSLATE THE TEXT TO TEXT BUFFER
                          ;
                          TRANS:
03DE 1E                          PUSH    DS
03DF 51                          PUSH    CX
03E0 56                          PUSH    SI
                          ;
03E1 16                          PUSH    SS
03E2 1F                          POP     DS
03E3 BE0202                      MOV     SI,OFFSET OUT_BUF
03E6 C60420                      MOV     BYTE PTR [SI],' '
03E9 46                          INC     SI
03EA 33C9                        XOR     CX,CX    ; CLEAR COUNTER
                          TRANS10:
03EC AC                          LODSB   ; COUNT LENGTH OF LINE.
03ED 41                          INC     CX
03EE 3C00                        CMP     AL,00H
03F0 75FA          03EC          JNE     TRANS10
03F2 BF4D00                      MOV     DI,4DH   ; ONE LINE TRANSLATE
03F5 E865FF        035D          CALL    ROM
                          ;
03F8 5E                          POP     SI
03F9 59                          POP     CX
03FA 1F                          POP     DS
03FB C3                          RET
                          ;
                          ;   CHECK NUMBER
                          ;
                          ;    INPUT AL
                          ;   OUTPUT CF=1 THEN NUMBER
                          ;          CF=0 THEN NON NUMBER
                          ;
                          CKNUMB:
03FC 3C30                        CMP     AL,'0'   ; AL<'0' ?
03FE 7203          0403          JB      CKNUMB10
0400 3C3A                        CMP     AL,'9'+1; AL>'9' ?
0402 C3                          RET
                          CKNUMB10:
0403 F5                          CMC
0404 C3                          RET
                          ;
                          ;   SKIP LINE NUMBER
                          ;
                          SKIP_LINE_NO:
0405 268A05                      MOV     AL,ES:[DI]
0408 47                          INC     DI
0409 E8F0FF        03FC          CALL    CKNUMB
040C 72F7          0405          JB      SKIP_LINE_NO
040E C3                          RET
                          ;
                                 END
```

15-3 バリアブルリスト(変数名リスト)

RAM上にLOADされている中間言語状態のプログラムのバリアブル リストをとるユーティリティーを紹介しましょう。

バリアブル リストとは，プログラム中で使用されている変数名（バリアブル ネーム）をすべて拾い出し，アルファベット順にソートして，何行で使われているかを表示するユーティリティープログラムのことです。

本プログラムにおいては，変数名だけではなく，ラベル，DEF FN関数もリストアップします。したがって，ラベルの相互参照のチェックにも使えます。

配列には，最後に'('をつけて表示します。

プログラムは，ROM内インタ プリタ コールの「テキストから1項目抽出」DI＝36 H(INT C4H コール)を使ってます。この1項目抽出ルーチンは，出力が，AL＝0 CHのときが変数名となっていますので，このとき，テキストエリアのプログラムから変数名リストテーブルへ登録するようにしています。このため，OUTPUTやBFやASも変数名とみてしまいますが，実用上は支障はないと思います。

1項目抽出ルーチンは，この他にもROM文や行番号の抽出にも使えます。利用価値は高いと思いますから，変数名リストのソースを参考にして使って下さい。

変数名テーブルのセグメントは，LTOP_SEG(オフセット2E1H)に入れてあります。

変数名テーブルがいっぱいになるとOut of memoryのエラーが出ます。

デフォールトでは，1700 Hに設定してありますので，2000 H-1700 Hの16倍約36 Kバイトあります。これで，560ケの変数名を登録できますが，メモリを拡張されている方は，LMAX(オフセット2E5H)の内容を8FB6Hから，FFB5Hに変更しますと変数名テーブルの大きさは約64 Kバイトになり約1000個の変数名を登録できます。

次にシステムディスクに入っている〝xfiles. n 88〟のバリアブル・リストのサンプル出力を示します。

使い方は、本プログラムを実行したのち変数名のリストをとりたいプログラムをロードします。
そして、DEF SEG＝&H1B00：A＝0：CALL Aとして下さい。

なお，プリンタに出力するには，

LLIST 0

として，すぐにこのルーチンを

A＝0：CALL A

などとして呼び出せばよいのです。

これは，LISTルーチンで〝カレントデバイスへの出力〟(INT C4H DI＝25 H)を使っているためです。CRT表示に戻すには、LIST 0とします。

```
0 ´SAVE"VALIST.BAS"
100 ´**************************************
110 ´*                                    *
120 ´*          VARIABLE LIST             *
130 ´*                                    *
140 ´*        FOR TechKnow9800            *
150 ´*                                    *
160 ´*     ALL RIGHTS & COPYRIGHT         *
170 ´*  RESERVED 1983 BY SYSTEMSOFT (C)   *
180 ´*                                    *
190 ´*  USAGE : CALL A                    *
200 ´*   MAKE VARIABLE LIST OF TEXT IN RAM*
210 ´*                                    *
220 ´**************************************
230 ´
240 DIM TEST$(0),TEST%(0),TEST!(0),TEST#(0)
250 DEF FNTEST(0)=TEST%(0):GOTO *START
260 *START
270  DEF SEG=&H1B00:CLEAR ,&H1B00
280  FOR I=0 TO &H31F
290   READ A$:POKE I,VAL("&H"+A$)
300  NEXT
310 BEEP 1
320  PRINT "Following list is variable list of this program !"
330 BEEP 0
340 PRINT
350 A=0:CALL A
360 END
370 ´
10000 DATA 8C,C8,2E,A3,E1,02,B8,1E,03,2E,A3,DF,02,2E,A3,E3:´0000H
10010 DATA 02,33,C0,2E,C4,3E,DF,02,B9,00,48,F3,AB,36,A1,A4:´0010H
10020 DATA 06,2E,A3,EE,02,2E,8B,36,EE,02,AD,85,C0,75,03,E9:´0020H
10030 DATA 8E,00,03,C6,2D,02,00,2E,A3,EE,02,AD,2E,A3,E7,02:´0030H
10040 DATA 36,89,36,EA,06,2E,89,36,EA,02,BF,36,00,CD,C4,36:´0040H
10050 DATA 8B,36,EA,06,3C,00,74,CD,3C,0C,75,E4,2E,89,36,EC:´0050H
10060 DATA 02,2E,8B,36,EA,02,0E,07,BF,F0,02,B9,15,00,33,C0:´0060H
10070 DATA 57,F3,AB,5F,8A,44,FF,3C,2A,75,03,AA,EB,08,3C,A1:´0070H
10080 DATA 75,04,B8,46,4E,AB,A4,AC,32,E4,3A,E0,74,04,8B,C8:´0080H
10090 DATA F3,A4,AC,4E,3C,24,74,10,3C,25,74,0C,3C,21,74,08:´0090H
10100 DATA 3C,23,74,04,3C,28,75,0A,AA,46,AC,4E,3C,28,75,02:´00A0H
10110 DATA AA,46,BF,F0,02,E8,83,00,2E,8B,36,EC,02,E9,80,FF:´00B0H
10120 DATA E8,3A,01,BF,1E,00,CD,C4,CF,51,56,36,A2,02,02,BF:´00C0H
10130 DATA 25,00,B9,01,00,E8,00,02,5E,59,C3,B0,20,E8,E9,FF:´00D0H
10140 DATA E2,F9,C3,56,57,52,51,BB,02,02,BF,34,00,E8,E8,01:´00E0H
10150 DATA 36,A1,02,02,36,8B,1E,04,02,36,8B,16,06,02,50,53:´00F0H
10160 DATA 51,52,B8,06,00,2B,C1,8B,C8,E8,CF,FF,5A,59,5B,58:´0100H
10170 DATA 36,A3,02,02,36,89,1E,04,02,36,89,16,06,02,BF,25:´0110H
10180 DATA 00,E8,B4,01,59,5A,5F,5E,C3,56,57,51,50,BF,4B,00:´0120H
10190 DATA E8,A5,01,58,59,5F,5E,C3,16,1F,C3,2E,C5,36,DF,02:´0130H
10200 DATA 8B,D6,2E,3B,36,E3,02,72,05,E8,18,00,EB,EA,BF,F0:´0140H
10210 DATA 02,E8,73,00,85,C9,75,05,E8,50,00,73,DB,8B,F2,83:´0150H
10220 DATA C6,40,EB,DC,83,C6,40,2E,89,36,E3,02,2E,3B,36,E5:´0160H
10230 DATA 02,76,07,B0,07,33,FF,CD,C4,CB,83,EE,40,E8,24,00:´0170H
10240 DATA B9,40,00,33,C0,F3,AA,83,EF,40,2E,A1,E7,02,26,89:´0180H
10250 DATA 45,2C,BE,F0,02,B9,2C,00,AC,AA,0A,C0,74,02,E2,F8:´0190H
10260 DATA E8,01,00,C3,06,1E,07,1F,87,FE,C3,B9,0A,00,AD,0B:´01A0H
10270 DATA C0,74,0B,2E,3B,06,E7,02,74,0B,E2,F2,F9,C3,2E,A1:´01B0H
10280 DATA E7,02,89,44,FE,F8,C3,B9,2C,00,AC,0A,C0,74,0F,26:´01C0H
10290 DATA 3A,05,9F,47,9E,74,04,73,01,F9,C3,E2,ED,C3,26,8A:´01D0H
10300 DATA 05,47,0A,C0,75,06,03,F1,4E,33,C9,C3,F9,C3,06,1E:´01E0H
```

```
10310 DATA 07,1F,E8,D2,FF,06,1E,07,1F,C3,16,1F,C3,E8,29,FF:'01F0H
10320 DATA 0E,07,2E,C5,36,DF,02,2E,3B,36,E3,02,74,EC,B0,FF:'0200H
10330 DATA 2E,A2,F0,02,33,C0,2E,A2,E9,02,8B,D6,8B,FE,BE,F0:'0210H
10340 DATA 02,E8,CA,FF,72,0E,85,C9,74,0A,8B,F2,BF,F0,02,B9:'0220H
10350 DATA 2C,00,F3,A4,8B,F2,83,C6,40,2E,3B,36,E3,02,72,DA:'0230H
10360 DATA 2E,A0,F0,02,3C,FF,74,B2,2E,8B,36,DF,02,8B,D6,8B:'0240H
10370 DATA FE,BE,F0,02,E8,97,FF,85,C9,75,0B,52,E8,16,00,5A:'0250H
10380 DATA 8B,F2,B0,FF,88,04,8B,F2,83,C6,40,2E,3B,36,E3,02:'0260H
10390 DATA 72,DB,E9,88,FF,B9,2C,00,2E,80,3E,E9,02,00,75,27:'0270H
10400 DATA 2E,FE,06,E9,02,8B,F2,AC,0A,C0,74,1B,E8,3A,FE,E2:'0280H
10410 DATA F6,B0,20,E8,33,FE,B9,0A,00,AD,0B,C0,74,05,E8,42:'0290H
10420 DATA FE,E2,F6,E8,17,00,C3,B8,01,00,83,E9,1A,76,02,8B:'02A0H
10430 DATA C1,8B,C8,E8,25,FE,8B,F2,83,C6,2C,EB,D9,50,52,57:'02B0H
10440 DATA 56,51,B8,0D,0A,36,A3,02,02,B9,02,00,BF,25,00,E8:'02C0H
10450 DATA 06,00,59,5E,5F,5A,58,C3,1E,16,1F,CD,C4,1F,C3,00:'02D0H
10460 DATA 00,00,17,00,00,B6,8F,00,00,00,00,00,00,00,00,00:'02E0H
10470 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00:'02F0H
10480 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00:'0300H
10490 DATA 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00:'0310H
```

xfiles.n88のバリアブル・リスト出力サンプル

```
*COPY.FILE          1250  1330
*GET.FILE.NAME      1190  1270  1590  1710
*GET.SEC            1460  1510
*INIT               1170  1740
*SET.ATR            1260  1440
*SKIP               1620  1660
A                   1050
A$                  1150
A0$                 1530  1760
AS                  1340  1350  1760  1770  1780
ATR$                1520  1680
D$                  1470  1480  1490  1500  1520  1530
DIR0$               1630  1640  1650  1670  1680
F1                  1690  1700  1710
FA$                 1380  1770
FB$                 1380  1770
FD                  1100  1240  1630  1640  1650  1790
FDR$                1100  1220
FILE.NAME$          1210  1220  1230  1240  1500  1670  1690  1700
FROM.DIRTRK         1630  1640  1650  1790
FROM.FILE$          1220  1340
FROM.MAXTRK         1630  1640  1650  1790
FROM.SEC            1610  1630  1640  1650  1750
FROM.SLOT           1600  1610  1620  1670  1680  1750
OUTPUT              1350
P                   1500  1510  1520
REC                 1360  1400
TA$                 1380  1780
TB$                 1380  1780
TD                  1120  1240  1470  1480  1490  1540  1550  1560  1800
TDR$                1120  1230
TO.DIRTRK           1470  1480  1490  1540  1550  1560  1800
TO.FILE$            1230  1350
TO.MAXTRK           1470  1480  1490  1540  1550  1560  1800
TO.SEC              1450  1470  1480  1490  1510  1540  1550  1560  1750
Z                   1050
```

```
                 ;****************************************
                 ;*                                      *
                 ;*      VARIABLE LIST FOR PC-9801       *
                 ;*                                      *
                 ;*   1983 COPYRIGHT BY SYSTEMSOFT (C)   *
                 ;*                                      *
                 ;*                                      *
                 ;****************************************
                 ;
                           SSEG
                           ORG     202H
0202             DISP_BUF RS      128
                           ORG     6A4H
06A4             TXTOP    RS      2    ; TEXT TOP
                           ORG     6EAH
06EA             TXTPNT   RS      2    ; TEXT POINTER
                 ;
                           CSEG
                           ORG     0000H
                 ;
  00E8                     CMD      EQU      0E8H
  002C                     VAL_LEN  EQU      44 ; MAX VARIABLE LENGTH
  0040                     REC_LEN  EQU VAL_LEN+20 ; 2BYTES*10(LINE NO)
  00A1                     FUNC     EQU      0A1H    ; DEF FN
                 ;
                 VALIST:
0000 8CC8                  MOV      AX,CS            ; SAVE CODE SEGMENT
0002 2EA3E102              MOV      WORD PTR LTOP+2,AX
0006 B81E03                MOV      AX,OFFSET LABEL
0009 2EA3DF02              MOV      LTOP,AX ; SET LABEL TABLE TOP.
000D 2EA3E302              MOV      LPNT,AX ; INITIALIZE LABEL TABLE PONTER.
                 ;
0011 33C0                  XOR      AX,AX    ; CLEAR LABEL TABLE
0013 2EC43EDF02            LES      DI,DWORD PTR LTOP
0018 B90048                MOV      CX,4800H
001B F3AB                  REP      STOSW
                 ;
001D 36A1A406              MOV      AX,WORD PTR TXTOP
0021 2EA3EE02              MOV      NLINE,AX ; INITIALIZE TEXT POINTER.
                 NEXT_LINE:
0025 2E8B36EE02            MOV      SI,NLINE ; LOAD NEXT LINE ADDRESS.
002A AD                    LODSW             ; LOAD LINK POINTER.
002B 85C0                  TEST     AX,AX    ; LINK POINTER=0 ?
002D 7503         0032     JNE      NEXT_LINE10 ; YES THEN PROGRAM END.
002F E98E00       00C0     JMP      PRO_END
                 NEXT_LINE10:
0032 03C6                  ADD      AX,SI    ; NEXT LINK POINTER.
0034 2D0200                SUB      AX,2
0037 2EA3EE02              MOV      NLINE,AX
                 ;
003B AD                    LODSW             ; GET LINE NO.
003C 2EA3E702              MOV      LIN_NO,AX
                 VALIST10:
0040 368936EA06            MOV      WORD PTR TXTPNT,SI
0045 2E8936EA02            MOV      PNT,SI
004A BF3600                MOV      DI,36H   ; GET 1 ITEM IN TEXT.
004D CDC4                  INT      0C4H
004F 368B36EA06            MOV      SI,WORD PTR TXTPNT
0054 3C00                  CMP      AL,0     ; END OF LINE ?
0056 74CD         0025     JE       NEXT_LINE
```

```
0058 3C0C                      CMP     AL,0CH  ; VARIABLE ?
005A 75E4          0040        JNE     VALIST10 ; NO THEN LOOP.
                       ;
005C 2E8936EC02                MOV     PNT10,SI ; SAVE NEXT POINT.
0061 2E8B36EA02                MOV     SI,PNT   ; LOAD VARIABLE TOP.
0066 0E                        PUSH    CS
0067 07                        POP     ES
0068 BFF002                    MOV     DI,OFFSET BUF
006B B91500                    MOV     CX,21
006E 33C0                      XOR     AX,AX
0070 57                        PUSH    DI          ; CLEAR BUFFER
0071 F3AB                      REP     STOSW
0073 5F                        POP     DI
0074 8A44FF                    MOV     AL,-1[SI]
0077 3C2A                      CMP     AL,'*'   ; LABEL ?
0079 7503          007E        JNE     VALIST12
007B AA                        STOSB
007C EB08          0086        JMPS    VALIST20
                       VALIST12:
007E 3CA1                      CMP     AL,FUNC ; DEF FN FUNCTION ?
0080 7504          0086        JNE     VALIST20
0082 B8464E                    MOV     AX,'NF'
0085 AB                        STOSW
                       VALIST20:
0086 A4                        MOVSB              ; FIRST CHR
0087 AC                        LODSB              ; GET LEN(VAL)-1
0088 32E4                      XOR     AH,AH
008A 3AE0                      CMP     AH,AL    ; AL=0 ?
008C 7404          0092        JE      VALIST30
008E 8BC8                      MOV     CX,AX
0090 F3A4                      REP     MOVSB    ; MOV BUF VARIABLE NAME.
                       ;
                       VALIST30:
0092 AC                        LODSB   ; VARIABLE[$%!#(]
0093 4E                        DEC     SI
0094 3C24                      CMP     AL,'$'
0096 7410          00A8        JE      VALIST40
0098 3C25                      CMP     AL,'%'
009A 740C          00A8        JE      VALIST40
009C 3C21                      CMP     AL,'!'
009E 7408          00A8        JE      VALIST40
00A0 3C23                      CMP     AL,'#'
00A2 7404          00A8        JE      VALIST40
00A4 3C28                      CMP     AL,'('
00A6 750A          00B2        JNE     VALIST50
                       VALIST40:
00A8 AA                        STOSB
00A9 46                        INC     SI
00AA AC                        LODSB
00AB 4E                        DEC     SI
00AC 3C28                      CMP     AL,'('
00AE 7502          00B2        JNE     VALIST50
00B0 AA                        STOSB
00B1 46                        INC     SI
                       VALIST50:
00B2 BFF002                    MOV     DI,OFFSET BUF
00B5 E88300        013B        CALL    TOUROKU
                       ;
00B8 2E8B36EC02                MOV     SI,PNT10 ; LOAD NEXT POINT.
00BD E980FF        0040        JMP     VALIST10
```

```
                          PRO_END:
00C0 E83A01        01FD            CALL   SORT
00C3 BF1E00                        MOV    DI,1EH    ; DIRECT MODE
00C6 CDC4                          INT    0C4H
00C8 CF                            IRET             ; FOR FAIL SAFE
                          ;
                          ;    OUTPUT CHR IN AL.
                          ;
                          OUTCHR:
00C9 51                            PUSH   CX
00CA 56                            PUSH   SI
00CB 36A20202                      MOV    BYTE PTR DISP_BUF,AL
00CF BF2500                        MOV    DI,25H    ; OUTPUT TO CURRENT DEVICE
00D2 B90100                        MOV    CX,1      ; 1 CHARACTER
00D5 E80002        02D8            CALL   ROM
00D8 5E                            POP    SI
00D9 59                            POP    CX
00DA C3                            RET
                          ;
                          ;    PRINT SPC(CX);
                          ;      DISPLAY CX SPACES.
                          SPC:
00DB B020                          MOV    AL,' '
00DD E8E9FF        00C9            CALL   OUTCHR
00E0 E2F9          00DB            LOOP   SPC
00E2 C3                            RET
                          ;
                          ;    DISPLAY LINE NO.
                          ;
                          DISP_NO:
00E3 56                            PUSH   SI        ; SAVE REISITERS
00E4 57                            PUSH   DI
00E5 52                            PUSH   DX
00E6 51                            PUSH   CX
                          ;
00E7 BB0202                        MOV    BX,OFFSET DISP_BUF
00EA BF3400                        MOV    DI,34H    ; PRINT VAL(AX)
00ED E8E801        02D8            CALL   ROM
                          ;
00F0 36A10202                      MOV    AX,WORD PTR DISP_BUF    ; SAVE NUMBER
00F4 368B1E0402                    MOV    BX,WORD PTR DISP_BUF+2
00F9 368B160602                    MOV    DX,WORD PTR DISP_BUF+4
                          ;
00FE 50                            PUSH   AX
00FF 53                            PUSH   BX
0100 51                            PUSH   CX
0101 52                            PUSH   DX
0102 B80600                        MOV    AX,6      ; PRINT SPC(6-CX);
0105 2BC1                          SUB    AX,CX     ;
0107 8BC8                          MOV    CX,AX     ;
0109 E8CFFF        00DB            CALL   SPC       ;
010C 5A                            POP    DX
010D 59                            POP    CX
010E 5B                            POP    BX
010F 58                            POP    AX
                          ;
0110 36A30202                      MOV    WORD PTR DISP_BUF,AX    ; LOAD NUMBER
0114 36891E0402                    MOV    WORD PTR DISP_BUF+2,BX
0119 3689160602                    MOV    WORD PTR DISP_BUF+4,DX
                          ;
```

```
011E BF2500                   MOV     DI,25H
0121 E8B401         02D8      CALL    ROM
                   ;
0124 59                       POP     CX        ; LOAD RESISTERS
0125 5A                       POP     DX
0126 5F                       POP     DI
0127 5E                       POP     SI
0128 C3                       RET
                   ;
                   ;  STOP ESC KEY SENSE
                   ;
                   SENSE:
0129 56                       PUSH    SI
012A 57                       PUSH    DI
012B 51                       PUSH    CX
012C 50                       PUSH    AX
012D BF4B00                   MOV     DI,4BH
0130 E8A501         02D8      CALL    ROM
0133 58                       POP     AX
0134 59                       POP     CX
0135 5F                       POP     DI
0136 5E                       POP     SI
0137 C3                       RET
                   ;
                   TOUROKU_END:
0138 16                       PUSH    SS
0139 1F                       POP     DS
013A C3                       RET
                   ;
                   ;    TOUROKU LABEL NAME INTO TABLE
                   ;
                   TOUROKU:
013B 2EC536DF02               LDS     SI,DWORD PTR LTOP
                   TOURO10:
0140 8BD6                     MOV     DX,SI    ; SAVE POINTER
0142 2E3B36E302               CMP     SI,LPNT ; COMPARE LAST POINTER
0147 7205           014E      JNAE    TOURO15 ; IF OVER THEN
0149 E81800         0164      CALL    NEW_RECORD ; NEW RECORD.
014C EBEA           0138      JMPS    TOUROKU_END ; END.
                   TOURO15:                     ; ELSE
014E BFF002                   MOV     DI,OFFSET BUF ; CHECK LABEL NAME.
0151 E87300         01C7      CALL    COMPARE
0154 85C9                     TEST    CX,CX
0156 7505           015D      JNE     TOURO20 ; IF SAME THEN
0158 E85000         01AB      CALL    TUIKA   ; ADD INTO LINE NO.
015B 73DB           0138      JNB     TOUROKU_END
                   TOURO20:
015D 8BF2                     MOV     SI,DX    ; LOAD POINTER
015F 83C640                   ADD     SI,REC_LEN ; NEXT RECORD
0162 EBDC           0140      JMPS    TOURO10
                   ;
                   ;  NEW VARIABLE NAME THEN MAKE NEW RECORD
                   ;
                   NEW_RECORD:
0164 83C640                   ADD     SI,REC_LEN; SET LPNT NEW PNT.
0167 2E8936E302               MOV     LPNT,SI
016C 2E3B36E502               CMP     SI,LMAX ; LMAX:LPNT ?
0171 7607           017A      JBE     NEW10    ; IF < THEN
0173 B007                     MOV     AL,7     ; OUT OF MEMORY
0175 33FF                     XOR     DI,DI
0177 CDC4                     INT     0C4H
```

```
0179 CB                       RETF             ; FOR FAIL SAFE
                        NEW10:                 ;  ELSE
017A 83EE40                   SUB     SI,REC_LEN ; CLEAR 1 RECORD
                        ;
017D E82400     01A4          CALL    CHG      ; XCHG SI:DS,DI:ES
                        ;
0180 B94000                   MOV     CX,REC_LEN
0183 33C0                     XOR     AX,AX
0185 F3AA                     REP     STOSB
0187 83EF40                   SUB     DI,REC_LEN
018A 2EA1E702                 MOV     AX,LIN_NO ; SET LINE NO.
018E 2689452C                 MOV     ES:VAL_LEN[DI],AX ; SET VARIABLE NAME.
0192 BEF002                   MOV     SI,OFFSET BUF
0195 B92C00                   MOV     CX,VAL_LEN
                        NEW20:                 ; FOR REPEAT VARIABLE LENGTH.
0198 AC                       LODSB
0199 AA                       STOSB
019A 0AC0                     OR      AL,AL    ; DETECT END MARK ?
019C 7402       01A0          JE      NEW_END
019E E2F8       0198          LOOP    NEW20
                        NEW_END:
01A0 E80100     01A4          CALL    CHG      ; XCHG SI:DS,DI:ES
01A3 C3                       RET
                        ;
01A4 061E071F           CHG:  PUSH ES ! PUSH DS ! POP ES ! POP DS
01A8 87FE                     XCHG DI,SI
01AA C3                       RET
                        ;
                        ;   LINE NUMBER WO TUIKA SURU.
                        ;
                        TUIKA:
01AB B90A00                   MOV     CX,10 ; 10 LINE NO PAR 1 RECORD
                        TUIKA10:
01AE AD                       LODSW
01AF 0BC0                     OR      AX,AX    ; END MARK ?
01B1 740B       01BE          JE      TUIKA20
01B3 2E3B06E702               CMP     AX,LIN_NO ; ALREADY EXISTS ?
01B8 740B       01C5          JE      TUIKA30
01BA E2F2       01AE          LOOP    TUIKA10
01BC F9                       STC              ; CF=1
01BD C3                       RET
                        TUIKA20:
01BE 2EA1E702                 MOV     AX,LIN_NO
01C2 8944FE                   MOV     -2[SI],AX ; SET LINE NO.
                        TUIKA30:
01C5 F8                       CLC              ; CF=0
01C6 C3                       RET
                        ;
                        ;  COMPARE DS:SI-ES:DI STRINGS
                        ;
                        ;      OUTPUT : CX==0 SAME NAME
                        ;               CX<>0 DIFFERENT NAME
                        ;                CF=1 THEN SI < DI
                        ;                CF=0 THEN SI > DI
                        ;
                        COMPARE:
01C7 B92C00                   MOV     CX,VAL_LEN
                        CMP10:
01CA AC                       LODSB
01CB 0AC0                     OR      AL,AL
```

```
01CD 740F        01DE            JE      CMP40
01CF 263A05                      CMP     AL,ES:[DI]
01D2 9F                          LAHF
01D3 47                          INC     DI
01D4 9E                          SAHF
01D5 7404        01DB            JE      CMP20   ; CF=0 CX<>0
01D7 7301        01DA            JNB     CMP30
01D9 F9                          STC     ; CF=1 CX<>0
01DA C3               CMP30:     RET
01DB E2ED        01CA CMP20:     LOOP    CMP10
01DD C3                          RET     ; CF=0 CX=0
                      CMP40:
01DE 268A05                      MOV     AL,ES:[DI]
01E1 47                          INC     DI
01E2 0AC0                        OR      AL,AL
01E4 7506        01EC            JNE     CMP50
01E6 03F1                        ADD     SI,CX
01E8 4E                          DEC     SI
01E9 33C9                        XOR     CX,CX   ; CF=0 CX=0
01EB C3                          RET
                      CMP50:
01EC F9                          STC     ; CF=1 CX<>0
01ED C3                          RET
                      COMPARE_II:
01EE 06                          PUSH    ES      ; XCHG ES,DS
01EF 1E                          PUSH    DS
01F0 07                          POP     ES
01F1 1F                          POP     DS
01F2 E8D2FF      01C7            CALL    COMPARE
01F5 06                          PUSH    ES      ; XCHG ES,DS
01F6 1E                          PUSH    DS
01F7 07                          POP     ES
01F8 1F                          POP     DS
01F9 C3                          RET
                      SEND:
01FA 16                          PUSH    SS
01FB 1F                          POP     DS
01FC C3                          RET
                      ;
                      ;     SORT LABEL AND DISPLAY
                      ;
                      SORT:
01FD E829FF      0129            CALL    SENSE
0200 0E                          PUSH    CS
0201 07                          POP     ES
0202 2EC536DF02                  LDS     SI,DWORD PTR LTOP
0207 2E3B36E302                  CMP     SI,LPNT ; LABEL TABLE EXISTS ?
020C 74EC        01FA            JE      SEND    ; NO THEN END.
020E B0FF                        MOV     AL,0FFH         ; DUMMY MAX VALUE SET
0210 2EA2F002                    MOV     BYTE PTR BUF,AL
0214 33C0                        XOR     AX,AX           ; LABEL FLAG CLEAR
0216 2EA2E902                    MOV     LFLG,AL
                      SORT10:
021A 8BD6                        MOV     DX,SI   ; SAVE POINTER
021C 8BFE                        MOV     DI,SI
021E BEF002                      MOV     SI,OFFSET BUF
                      ;
0221 E8CAFF      01EE            CALL    COMPARE_II ; SI:DS-DI:ES
                      ;
0224 720E        0234            JB      SORT20  ;  BUF - TABLE
```

```
0226 85C9                    TEST    CX,CX
0228 740A          0234      JE      SORT20
022A 8BF2                    MOV     SI,DX    ; MOV LABEL NAME INTO BUFFER.
022C BFF002                  MOV     DI,OFFSET BUF
022F B92C00                  MOV     CX,VAL_LEN
                                              ; DI:ES<-SI:DS
                                              ; BUF  <- TABLE
0232 F3A4                    REP     MOVSB
                   SORT20:
0234 8BF2                    MOV     SI,DX    ; LOAD POINTER
0236 83C640                  ADD     SI,REC_LEN ; NEXT RECORD
0239 2E3B36E302              CMP     SI,LPNT  ; END RECORD ?
023E 72DA          021A      JB      SORT10   ; NO THEN LOOP
0240 2EA0F002                MOV     AL,BYTE PTR BUF
0244 3CFF                    CMP     AL,0FFH  ; END MARK ?
0246 74B2          01FA      JE      SEND     ; YES THEN END.
0248 2E8B36DF02              MOV     SI,LTOP  ; NO  THEN OUTPUT LABELS.
                   SORT30:                    ;         AND LINE NO.
024D 8BD6                    MOV     DX,SI    ; SAVE POINTER
024F 8BFE                    MOV     DI,SI
0251 BEF002                  MOV     SI,OFFSET BUF
                   ;
0254 E897FF        01EE      CALL    COMPARE_II
                   ;
0257 85C9                    TEST    CX,CX
0259 750B          0266      JNE     SORT40
025B 52                      PUSH    DX
025C E81600        0275      CALL    OUTPUT
025F 5A                      POP     DX
0260 8BF2                    MOV     SI,DX    ; LOAD POINTER
0262 B0FF                    MOV     AL,0FFH
0264 8804                    MOV     [SI],AL  ; OUTPUT MARK.
                   SORT40:
0266 8BF2                    MOV     SI,DX    ; LOAD POINTER
0268 83C640                  ADD     SI,REC_LEN ; NEXT RECORD
026B 2E3B36E302              CMP     SI,LPNT  ; END ?
0270 72DB          024D      JB      SORT30
0272 E988FF        01FD      JMP     SORT
                   ;
                   OUTPUT:
0275 B92C00                  MOV     CX,VAL_LEN ; MAX VARIABLE LENGTH
0278 2E803EE90200            CMP     LFLG,0
027E 7527          02A7      JNE     OUTPUT40
0280 2EFE06E902              INC     LFLG
0285 8BF2                    MOV     SI,DX    ; LOAD POINTER
                   OUTPUT10:                   ; OUTPUT LABEL NAME.
0287 AC                      LODSB
0288 0AC0                    OR      AL,AL     ; IF END MARK THEN END.
028A 741B          02A7      JE      OUTPUT40
028C E83AFE        00C9      CALL    OUTCHR
028F E2F6          0287      LOOP    OUTPUT10 ;
0291 B020                    MOV     AL,' '
0293 E833FE        00C9      CALL    OUTCHR
                   OUTPUT20:
0296 B90A00                  MOV     CX,10    ; 10 LINE NO IN ONE LINE.
                   OUTPUT25:
0299 AD                      LODSW            ; GET LINE NO.
029A 0BC0                    OR      AX,AX
029C 7405          02A3      JE      OUTPUT30
029E E842FE        00E3      CALL    DISP_NO
02A1 E2F6          0299      LOOP    OUTPUT25
```

```
                           OUTPUT30:
02A3 E81700        02BD              CALL     CRLF
02A6 C3                              RET
                           OUTPUT40:
02A7 B80100                          MOV      AX,1
02AA 83E91A                          SUB      CX,VAL_LEN-18
02AD 7602          02B1              JNA      OUTPUT50
02AF 8BC1                            MOV      AX,CX
                           OUTPUT50:
02B1 8BC8                            MOV      CX,AX
02B3 E825FE        00DB              CALL     SPC
                           ;
02B6 8BF2                            MOV      SI,DX    ; POINT LINE NO.
02B8 83C62C                          ADD      SI,VAL_LEN
02BB EBD9          0296              JMPS     OUTPUT20
                           ;
                           ;      OUTPUT CRLF TO CURRENT DEVICE
                           ;
                           CRLF:
02BD 50                              PUSH     AX
02BE 52                              PUSH     DX
02BF 57                              PUSH     DI
02C0 56                              PUSH     SI
02C1 51                              PUSH     CX
02C2 B80D0A                          MOV      AX,0A0DH
02C5 36A30202                        MOV      WORD PTR DISP_BUF,AX
02C9 B90200                          MOV      CX,2
02CC BF2500                          MOV      DI,25H
02CF E80600        02D8              CALL     ROM
02D2 59                              POP      CX
02D3 5E                              POP      SI
02D4 5F                              POP      DI
02D5 5A                              POP      DX
02D6 58                              POP      AX
02D7 C3                              RET
                           ;
                           ROM:
02D8 1E                              PUSH     DS        ; SAVE DS
02D9 16                              PUSH     SS        ; DS=60H
02DA 1F                              POP      DS        ; AND CALL ROM.
02DB CDC4                            INT      0C4H      ;
02DD 1F                              POP      DS        ; LOAD DS
02DE C3                              RET
                           ;
                           ;****************** WORK AREA ******************
                           ;
02DF 0000                  LTOP     DW       0000H     ; LABEL TABLE TOP
02E1 0017                  LTOP_SEG DW       1700H     ; ITS SEGMENT
02E3 0000                  LPNT     DW       0000H     ; LABEL TABLE POINTER
02E5 B68F                  LMAX     DW 9000H-REC_LEN-10 ; LABEL TABLE MAX OFFSET
02E7 0000                  LIN_NO   DW       0000H     ; LINE NO SAVE AREA
02E9 00                    LFLG     DB       00H       ; LABEL FLAG
                           ;
02EA 0000                  PNT      DW       0000H     ; LAST TEXT POINT
02EC 0000                  PNT10    DW       0000H     ; TEXT POINTER BUFFER
02EE 0000                  NLINE    DW       0000H     ; NEXT LINE POINTER
                           ;
02F0                       BUF      RS       VAL_LEN   ; LABEL NAME BUFFER
031C 0000                           DW       0000H     ; FOR FAIL SAFE
                           ;
```

```
                         ;=================== LABEL TABLE ==================
                         ;
031E                     LABEL   RS    9000H
                         ;
                         ;=================================================

                                 END
```

15-4 バーティカル・ファイルズ

ディスクのファイルをロードする際，よくFILESを実行して，ファイル名を表示しますね。これですとファイルが横に連なって，区切りがないため，カーソルをロードしたいところへもっていってLOAD″とやってもSyntax errorとむなしく表示されるだけです。

次のプログラムを実行すると，例のようにファイル名が1つずつ縦に表示されます。しかも，"マークとドライブNO.も先頭に付きますのでロードまたは実行したいファイルのところにカーソルを移動させ，LOADやRUNとするだけでファイルのロードができます。

FILESを実行すると，ファイル名を15個表示して止まります(15個に未たないときはそれらを表示してOKが表示されます)。目的のファイルがあればカーソルキーの[↑]を押します。ファイルの続きを表示するには[↓]キーを押します。なお，このプログラムは，拡張コマンド解析，スペーススキップ，式解析，ディスク LIO，CRT1文字出力，エラーメッセージ出力，キーセンスなどのルーチンを利用しています。

なお，これは全ドライブ対応となっており，ディスクユニットがいくつ接続されていてもかまいません。

＜実行例＞

```
FILES 2
      "2:Tfiles.n98
      "2:Finfo .n98
      "2:D-edit.98
      "2:Kanji .key
      "2:TSS-V2.0
      "2:kanji .bas
      "2:cursor.
      "2:COPY  .MAC
      "2:PkanjI.
      "2:Kdisp .
      "2:TXT   .BAS
      "2:kdisp .bas
      "2:VRAM  .bas
      "2:HCLS  .BAS
      "2:DIR   .SRT
Ok
```

バーティカル・ファイルズ

```
1 ´save "Vfiles.n88" : ´ Ver. 2.1
100 CLEAR ,&H1F00:DEF SEG=&H1F00
110 CLS :PRINT "=== VERTICAL FILES for All Drives ==="
120 RESTORE 340
130 FOR I=0 TO &HE5
140  READ D$:POKE I,VAL("&H"+D$)
150 NEXT I
160  ´
170 DEF SEG=&H60
180 FL=&H1593   ´ -- Extended Command Flag
190 AD=&H159C
200 POKE FL,0:RESTORE 290
210 FOR I=0 TO 7 ´ -- Set User Routine Address
220  READ D$:D=VAL("&H"+D$)
230  POKE AD+I,D
240 NEXT I
250 POKE FL,1  ´ -- Flag On
260 PRINT "Complete! Now, you can get VERTICAL FILES."
270  END
280 ´ User Routine Add
290 DATA 00,00,00,1F  ´ V-FILES
300 DATA 07,00,00,1F  ´ RETF
310 ´-----------------------------------------------
320 ´ VERTICAL FILES Machine Code
330 ´-----------------------------------------------
340 DATA 16,1F,3C,A0,F8,74,0B,CB,B0,46,EB,02,B0,05,33,FF
350 DATA CD,C4,56,89,36,EA,06,BF,22,00,CD,C4,89,36,EA,06
360 DATA AC,5E,3C,00,74,20,3C,3A,74,1C,89,36,EA,06,BF,4A
370 DATA 00,CD,C4,A0,1A,14,8A,26,03,05,FE,C4,3A,C4,73,C8
380 DATA 3C,00,76,C8,EB,02,B0,01,8A,F0,32,D2,16,07,B4,03
390 DATA 8B,1E,0A,05,C6,47,06,FF,50,CD,B0,0A,E4,75,14,53
400 DATA E8,22,00,5B,C6,47,06,00,58,80,FA,0F,75,03,E8,5A
410 DATA 00,EB,E5,5B,80,FC,37,74,04,8A,C4,EB,91,A1,EA,06
420 DATA A3,E8,06,F9,CB,FE,C2,B9,05,00,B0,20,E8,36,00,E2
430 DATA F9,B0,22,E8,2F,00,8A,C6,0C,30,E8,28,00,B0,3A,E8
440 DATA 23,00,B9,06,00,8A,47,06,E8,1A,00,43,E2,F7,B0,2E
450 DATA E8,12,00,B9,03,00,8A,47,06,E8,09,00,43,E2,F7,BF
460 DATA 40,00,CD,C4,C3,BF,3D,00,CD,C4,C3,32,D2,50,B4,04
470 DATA B0,07,CD,18,F6,C4,04,75,07,F6,C4,20,75,06,EB,EE
480 DATA 58,58,EB,99,58,C3,00,00,00,00,00,00,00,00,00,00
```

バーティカル・ファイルズ・ソースリスト

```
                          ;========================================
                          ; Vertical FILES for All Drives
                          ; 8 INCH - 5 INCH 2D - 5 INCH 2DD
                          ; COMMAND: FILES n ( n = Drive No.)
                          ; Enhanced 1984.2.3 by E.F.
                          ;========================================
                          ;
 1F00                              CSEG 1F00H
                                   ORG 0
                          ;
 00A0                     FILES    EQU 0A0H
 06EA                     EXEC     EQU 6EAH
 06E8                     NEXT     EQU 6E8H
 141A                     RES      EQU 141AH
 050A                     FCB      EQU 50AH        ; File Control Block
 0503                     DISKNO   EQU 503H        ; Total Disk Drive No.
                          ;
0000 16                            PUSH SS
0001 1F                            POP DS          ; DS<=60H
0002 3CA0                 CHECK:   CMP AL,FILES
0004 F8                            CLC
0005 740B          0012            JE SPACE
0007 CB                            RETF
                          ;------------------------------------------------
0008 B046                 BADDR:   MOV AL,46H      ; Bad Drive No.
000A EB02          000E            JMPS ERR
000C B005                 ILLEGAL:MOV AL,5         ; Illegal function call
000E 33FF                 ERR:     XOR DI,DI       ; DISK error
0010 CDC4                          INT 0C4H
                          ;------------------------------------------------
0012 56                   SPACE:   PUSH SI
0013 8936EA06                      MOV .EXEC,SI    ; SPACE SKIP
0017 BF2200                        MOV DI,22H
001A CDC4                          INT 0C4H
001C 8936EA06                      MOV .EXEC,SI    ; Store Text Pointer
0020 AC                            LODSB
0021 5E                            POP SI
0022 3C00                          CMP AL,0        ; FILES
0024 7420          0046            JE DRIVE1
0026 3C3A                          CMP AL,3AH      ; ':' ? FILES : check
0028 741C          0046            JE DRIVE1
                          ;
002A 8936EA06                      MOV .EXEC,SI    ; EXPRESSION ANALYSIS
002E BF4A00                        MOV DI,4AH
0031 CDC4                          INT 0C4H
0033 A01A14                        MOV AL,.RES     ; RESULT
0036 8A260305                      MOV AH,.DISKNO  ; Max Drive No.
003A FEC4                          INC AH
003C 3AC4                          CMP AL,AH       ; BAD DRIVE NO.
003E 73C8          0008            JAE BADDR
0040 3C00                          CMP AL,0
0042 76C8          000C            JBE ILLEGAL
0044 EB02          0048            JMPS STORE
0046 B001                 DRIVE1:  MOV AL,1        ; Drive 1
0048 8AF0                 STORE:   MOV DH,AL       ; DH <= Drive No.
                          ;
004A 32D2                          XOR DL,DL       ; DL = Counter
004C 16                   DIRGET:  PUSH SS
```

```
004D 07                       POP ES             ; ES <= 60H
004E B403                     MOV AH,3           ; DISK LIO BIOS CODE
0050 8B1E0A05                 MOV BX,.FCB
0054 C64706FF                 MOV BYTE PTR 6[BX],0FFH ; FIRST PROCESS
                      ;
0058 50               FILELP: PUSH AX
0059 CDB0                     INT 0B0H           ; GET DIRECTORY
005B 0AE4                     OR AH,AH
005D 7514        0073         JNE ERROR
                      ;
005F 53                       PUSH BX
0060 E82200      0085         CALL FILED         ; Filename Display
0063 5B                       POP BX
0064 C6470600                 MOV BYTE PTR 6[BX],00H  ; NEXT PROCESS
0068 58                       POP AX
0069 80FA0F                   CMP DL,15          ; 15 FILES DISPLAY
006C 7503        0071         JNE NXT
006E E85A00      00CB         CALL KEYSEN        ; WAIT FOR KEYIN
0071 EBE5        0058 NXT:    JMPS FILELP
                      ;
0073 5B               ERROR:  POP BX             ; Dummy POP
0074 80FC37                   CMP AH,55          ; Directory End
0077 7404        007D         JE PEND
0079 8AC4                     MOV AL,AH          ; Else DISK error
007B EB91        000E         JMPS ERR
                      ;
007D A1EA06           PEND:   MOV AX,.EXEC
0080 A3E806                   MOV .NEXT,AX
0083 F9                       STC
0084 CB               PBACK:  RETF               ; END OF MAIN
                      ;
0085 FEC2             FILED:  INC DL             ; COUNTER INCREMENT
0087 B90500                   MOV CX,5           ; _____"DNO.:
008A B020             LP2:    MOV AL,20H
008C E83600      00C5         CALL DISP
008F E2F9        008A         LOOP LP2
                      ;
0091 B022                     MOV AL,'"'
0093 E82F00      00C5         CALL DISP
0096 8AC6                     MOV AL,DH          ; DRIVE NO.
0098 0C30                     OR AL,30H          ; Convert to ASCII
009A E82800      00C5         CALL DISP
009D B03A                     MOV AL,':'
009F E82300      00C5         CALL DISP
                      ;
00A2 B90600                   MOV CX,6
00A5 8A4706           LP3:    MOV AL,6[BX]       ; Filename Display
00A8 E81A00      00C5         CALL DISP
00AB 43                       INC BX
00AC E2F7        00A5         LOOP LP3
                      ;
00AE B02E                     MOV AL,'.'
00B0 E81200      00C5         CALL DISP
                      ;
00B3 B90300                   MOV CX,3
00B6 8A4706           LP4:    MOV AL,6[BX]
00B9 E80900      00C5         CALL DISP
00BC 43                       INC BX
00BD E2F7        00B6         LOOP LP4
```

```
                        ;
00BF BF4000             CRLF:   MOV DI,40H      ; CR/LF INT CALL
00C2 CDC4                       INT 0C4H
                        ;
00C4 C3                 BACK:   RET
                        ;
00C5 BF3D00             DISP:   MOV DI,3DH      ; ONE CHR DISPLAY
00C8 CDC4                       INT 0C4H
00CA C3                         RET
                        ;
00CB 32D2               KEYSEN: XOR DL,DL
00CD 50                         PUSH AX
00CE B404               KEY:    MOV AH,04H      ; KEY SENSE
00D0 B007                       MOV AL,07H
00D2 CD18                       INT 18H
00D4 F6C404                     TEST AH,04H     ; [CURSOR UP]
00D7 7507         00E0          JNE UP
00D9 F6C420                     TEST AH,20H     ; [CURSOR DOWN]
00DC 7506         00E4          JNE DOWN
00DE EBEE         00CE          JMPS KEY
                        ;
00E0 58                 UP:     POP AX          ; Dummy
00E1 58                         POP AX          ; STACK BACK
00E2 EB99         007D          JMPS PEND       ; BACK TO BASIC
                        ;
00E4 58                 DOWN:   POP AX          ; Restore AX
00E5 C3                         RET             ; RET TO MAIN
                        ;
                        ;----------------------------------------
                        ;
                                END
```

付　録

付録－1　マシン語プログラム・ソースリスト

付録－2　ROM内ルーチンのＩＮＴによる利用

(1) ＩＮＴ割り込みベクター覧表

(2) ＩＮＴ C4Hのソフトウェア インターフェイスの説明

（インタプリタ内のルーチンの利用）

付録－3　ワークエリアー覧表

(1) システム共通域

(2) ＢＡＳＩＣ　ＬＩＯワークエリア

(3) シンボルテーブルのワークエリア

付録－4　Ｉ／Ｏポート一覧表

付録－5　コマンド・ステートメント・関数処理

アドレス一覧表

付録－6　コントロールコード一覧表

付録－7　エラーメッセージ一覧表

付録－8　プリンタ機能一覧表

（ＰＣ－8821／22・ＰＣ－8023

・ＮＫ－3618－21／22）

付録－9　キャラクタコード表

付録－10　ＵＳＩＮＧ文フォーマット一覧表

付録－11　Ｚ－80・8086ニーモニック対応表

付録－1

マシン語プログラム・ソースリスト

付録1　マシン語プログラム・ソースリスト

本文中のマシン語のプログラムは解説のためにソースリストもあわせて示していますが、ここでは、BASICのDATA文中にあるマシン語プログラムのソースリストを掲載します。8086のアセンブリ言語でのプログラミングやPC-9801の解析などの参考になれば幸いです。具体的な使い方・内容は、各章を参照して下さい。

1．テキスト画面の２ページ目を利用
2．マシン語によるG-VRAM直接アクセス法
3．ファンクションキー退避・復活
4．インターリーブ13フォーマット
5．8インチIDリーダー
6．テキスト画面コピー
7．グラフィック画面コピー
8．PRINT/LPRINT (CALL文のルーチン)
9．PRINT/LPRINT (CMD ON/OFF)
10．漢字フォントをビットイメージで出力

1．テキスト画面の２ページ目を利用

```
                         ;====================================
                         ;*  USE TWO TEXT VRAMS              *
                         ;*  Calling sequence ...            *
                         ;*  DEF SEG=&H1F00 : VRAM=&H0       *
                         ;*  A%=0:CALL VRAM(A%) ' Transfer   *
                         ;*  A%=1:CALL VRAM(A%) ' Restore    *
                         ;====================================
                         ;
 A000                    VRAM1    EQU 0A000H      ; TEXT VRAM1 Segment
 A100                    VRAM2    EQU 0A100H      ; TEXT VRAM2 Segment
 A200                    ATTR1    EQU 0A200H      ; Attribute1 Segment
 A300                    ATTR2    EQU 0A300H      ; Atrribute2 Segment
 07FF                    NWORD    EQU 07FFH       ; No.of Words to Transfer
                         ;
0000 C437                GETP:    LES SI,[BX]     ; Get parameter form
0002 268B04                       MOV AX,ES:[SI]  ; BASIC CALL ' AX=A%
0005 3D0000                       CMP AX,0
0008 7406       0010              JE TRANS        ; IF A%=0 THEN TRANSFER
000A 3D0100                       CMP AX,1
000D 7414       0023              JE RESTR        ; IF A%=1 THEN RESTORE
000F CF                  BACK:    IRET            ; IF NONE THEN BASIC
                         ;
0010 B800A0              TRANS:   MOV AX,VRAM1    ; VRAM1 --> VRAM2
0013 BB00A1                       MOV BX,VRAM2
0016 E81D00     0036              CALL BLOCKT     ; BLOCK Transfer
0019 B800A2                       MOV AX,ATTR1    ; ATTR1 --> ATTR2
001C BB00A3                       MOV BX,ATTR2
001F E81400     0036              CALL BLOCKT
0022 CF                           IRET            ; Back to BASIC
                         ;
0023 B800A1              RESTR:   MOV AX,VRAM2    ; VRAM2 --> VRAM1
0026 BB00A0                       MOV BX,VRAM1
0029 E80A00     0036              CALL BLOCKT
002C B800A3                       MOV AX,ATTR2    ; ATTR2 --> ATTR1
002F BB00A2                       MOV BX,ATTR1
0032 E80100     0036              CALL BLOCKT
0035 CF                           IRET            ; Back to BASIC
                         ;
0036 8ED8                BLOCKT:  MOV DS,AX       ; Source Segment
0038 8EC3                         MOV ES,BX       ; Destination Segment
003A 33F6                         XOR SI,SI       ; Source Offset
003C 33FF                         XOR DI,DI       ; Destination Offset
003E FC                           CLD             ; Increment Mode
003F B9FF07                       MOV CX,NWORD    ; No.of Words
0042 F3A5                         REP MOVSW       ; Move String Word
0044 C3                           RET
                         ;
                                  END
```

2. マシン語によるG-VRAM直接アクセス法

```
                        ;==========================================
                        ;* High Speed and RANDOM CLS              *
                        ;*  Calling sequence ...                  *
                        ;*  DEF SEG=&H1F00 : HCLS=0               *
                        ;*  A%=0:B%=0          ' All clear        *
                        ;*  A%=1:B%=1 or 2 or 3 ' Random clear    *
                        ;*  CALL HCLS(A%,B%)                      *
                        ;==========================================
                        ;
 A800                   BLUE     EQU 0A800H
 B000                   RED      EQU 0B000H
 B800                   GREEN    EQU 0B800H
 00A0                   PORT1    EQU 0A0H
 00A2                   PORT2    EQU 0A2H
                        ;
                                 CSEG
                                 ORG 0
                        ;
0000 C47704             GPARA:   LES SI,4[BX]      ; GET PAPAMETERS
0003 268B04                      MOV AX,ES:[SI]    ; AX=A%
                        ;
0006 8B37                        MOV SI,[BX]
0008 268B1C                      MOV BX,ES:[SI]    ; BX=B%
                        ;
000B 3D0000             COMP:    CMP AX,0          ; MODE SELECT
000E 7406         0016           JE MODE0          ; ALL CLEAR
0010 3D0100                      CMP AX,1
0013 742D         0042           JE MODE1          ; RANDOM CLEAR
                        ;
0015 CF                          IRET              ; BACK TO BASIC
                        ;
                        ;
0016 E8EF00       0108  MODE0:   CALL WAIT         ; GDC CHECK
                        ;
0019 B00C               OFF:     MOV AL,0CH        ; DISPLAY OFF
001B E6A2                        OUT PORT2,AL
                        ;
001D B800A8             ALL:     MOV AX,BLUE       ; CLEAR BLUE
0020 E81100       0034           CALL CLS
0023 B800B0                      MOV AX,RED        ; CLEAR RED
0026 E80B00       0034           CALL CLS
0029 B800B8                      MOV AX,GREEN      ; CLEAR GREEN
002C E80500       0034           CALL CLS
                        ;
002F B00D               ON:      MOV AL,0DH        ; DISPLAY ON
0031 E6A2                        OUT PORT2,AL
                        ;
0033 CF                 BACK:    IRET              ; BACK TO BASIC
                        ;
                        ;
0034 B9FF3F             CLS:     MOV CX,3FFFH      ; 32K CLEAR
0037 8EC0                        MOV ES,AX
0039 BF0000                      MOV DI,0
003C 33C0                        XOR AX,AX
003E FC                          CLD
003F F2AB                        REPNE STOSW
0041 C3                          RET               ; RETURN TO ALL
                        ;
                        ;
```

```
0042 E8C300        0108 MODE1:  CALL WAIT
0045 83FB01                     CMP BX,1        ; RANDOM CLEAR
0048 740B          0055         JE CURT         ; CURTAIN CLEAR
004A 83FB02                     CMP BX,2
004D 7444          0093         JE SHUT         ; SHUTTLE CLEAR
004F 83FB03                     CMP BX,3
0052 7473          00C7         JE BUB          ; BUBBLE CLEAR
                        ;
0054 CF                         IRET            ; IF NONE THEN RETURN
                        ;
0055 B800A8             CURT:   MOV AX,BLUE     ; CURTAIN CLEAR
0058 E80D00        0068         CALL CTSUB
005B B800B0                     MOV AX,RED
005E E80700        0068         CALL CTSUB
0061 B800B8                     MOV AX,GREEN
0064 E80100        0068         CALL CTSUB
0067 CF                         IRET            ; RETURN TO BASIC
0068 B307               CTSUB:  MOV BL,7        ; CURT SUB
006A E80B00        0078         CALL CTSUB1
006D B303                       MOV BL,3
006F E80600        0078         CALL CTSUB1
0072 B300                       MOV BL,0
0074 E80100        0078         CALL CTSUB1
0077 C3                         RET             ; RETURN TO CURT
0078 8ED8               CTSUB1: MOV DS,AX       ; CURT SUB1
007A 33FF                       XOR DI,DI
007C B95000                     MOV CX,50H      ; 80 LINES
007F 51                 L80:    PUSH CX
0080 57                         PUSH DI
0081 B99001                     MOV CX,190H     ; 400 DOTS
0084 881D               L400:   MOV [DI],BL
0086 83C750                     ADD DI,50H
0089 E0F9          0084         LOOPNE L400
008B 5F                         POP DI
008C 83C701                     ADD DI,1H
008F 59                         POP CX
0090 E0ED          007F         LOOPNE L80
0092 C3                         RET             ; RETURN TO CTSUB
                        ;
0093 B800A8             SHUT:   MOV AX,BLUE     ; SHUTTLE CLEAR
0096 E80D00        00A6         CALL SSUB1
0099 B800B0                     MOV AX,RED
009C E80700        00A6         CALL SSUB1
009F B800B8                     MOV AX,GREEN
00A2 E80100        00A6         CALL SSUB1
00A5 CF                         IRET            ; RETURN TO BASIC
                        ;
00A6 8ED8               SSUB1:  MOV DS,AX       ; SHUT SUB1
00A8 33DB                       XOR BX,BX
00AA B9FF7C                     MOV CX,7CFFH
00AD E81200        00C2 NDONE:  CALL SSUB2
00B0 87CB                       XCHG CX,BX
00B2 E80D00        00C2         CALL SSUB2
00B5 87CB                       XCHG CX,BX
00B7 43                         INC BX
00B8 43                         INC BX
00B9 49                         DEC CX
00BA 49                         DEC CX
00BB 8AC5                       MOV AL,CH
00BD 3C7F                       CMP AL,7FH
00BF 75EC          00AD         JNE NDONE
```

```
00C1 C3                         RET                  ; RETURN TO SHUT
00C2 C6470100          SSUB2:   MOV BYTE [BX],0      ; SHUT SUB2
00C6 C3                         RET                  ; RETURN TO SSUB1
                       ;
00C7 B9F51E            BUB:     MOV CX,7925          ; CX = PATTERN
00CA B800A8                     MOV AX,BLUE
00CD E80D00      00DD           CALL BUB1
00D0 B800B0                     MOV AX,RED
00D3 E80700      00DD           CALL BUB1
00D6 B800B8                     MOV AX,GREEN
00D9 E80100      00DD           CALL BUB1
00DC CF                         IRET                 ; RETURN TO BASIC
                       ;
00DD 8ED8              BUB1:    MOV DS,AX
00DF 33DB                       XOR BX,BX
00E1 33D2                       XOR DX,DX
00E3 51                         PUSH CX
00E4 D0FD              SHIFT:   SAR CH,1
00E6 D0D9                       RCR CL,1
00E8 7207        00F1           JB BELOW
00EA D0EE                       SHR DH,1
00EC D0DA                       RCR DL,1
00EE E9F3FF      00E4           JMP SHIFT
00F1 59                BELOW:   POP CX
00F2 52                CONT:    PUSH DX
00F3 8AC7                       MOV AL,BH
00F5 0C00                       OR AL,00
00F7 8AF8                       MOV BH,AL
00F9 C6470100                   MOV BYTE [BX],0
00FD 03D9                       ADD BX,CX
00FF 5A                         POP DX
0100 4A                         DEC DX
0101 8AC6                       MOV AL,DH
0103 0AC2                       OR AL,DL
0105 75EB        00F2           JNE CONT
0107 C3                         RET                  ; RETURN TO BUB
                       ;
                       ;-------------------------------------------
                       ;        GDC STATUS CHECK
                       ;-------------------------------------------
                       WAIT:
0108 E4A0              EMPTY:   IN AL,PORT1          ; FIFO EMPTY CHECK
010A A804                       TEST AL,04H
010C 74FA        0108           JZ EMPTY
                       ;
010E E4A0              VSYNC:   IN AL,PORT1          ; VSYNC CHECK
0110 A820                       TEST AL,20H
0112 74FA        010E           JZ VSYNC
                       ;
0114 E4A0              NDRAW:   IN AL,PORT1
0116 A808                       TEST AL,08H
0118 75FA        0114           JNZ NDRAW
                       ;
011A C3                         RET
                       ;
                                END
```

3. ファンクションキー退避・復活

```
                          ;===================================
                          ;  Function Key Save & Restrore
                          ;   Calling sequence...
                          ;   DEF SEG=&H1F00
                          ;   FK=0
                          ;   A%=0:CALL FK(A%) ' Save
                          ;   A%=1:CALL FK(A%) ' Restore
                          ;===================================
                          ;
 1F00                            CSEG 1F00H
                          ;
 0378                     FUNCKY EQU 0378H
 005A                     LEN    EQU 90          ; 180 bytes=90 words
                          ;
                                 ORG 0000H
                          ;
0000 C537                 CALL:  LDS SI,[BX]
0002 8B04                        MOV AX,[SI]     ; AX=A%
                          ;
0004 16                          PUSH SS
0005 5B                          POP BX          ; BX=60H
0006 0E                          PUSH CS
0007 59                          POP CX          ; CX=1F00H
0008 0E                          PUSH CS
0009 1F                          POP DS          ; DS=1F00H
000A BE7803                      MOV SI,FUNCKY
000D BF2300                      MOV DI,OFFSET FKBUF
                          ;
0010 0BC0                        OR AX,AX
0012 7404           0018         JZ SAVE         ; IF AX=0 THEN SAVE
0014 87D9                 RESTR: XCHG BX,CX
0016 87F7                        XCHG SI,DI
                          ;
0018 8EDB                 SAVE:  MOV DS,BX
001A 8EC1                        MOV ES,CX
001C FC                          CLD
001D B95A00                      MOV CX,LEN
0020 F3A5                        REP MOVSW       ; Word ransfer
0022 CF                          IRET            ; Back to BASIC
                          ;
0023                      FKBUF  RW 90           ; Key Buffer
                          ;
                                 END
```

4. インターリーブ13フォーマット

```
                          ;=============================================
                          ;  Interleave 13 Formatting
                          ;  CALL FORM13(STS,CYL,SF,SECLEN)
                          ;   SECLEN=0 OR 1 (0:128 BYTE,1:256 BYTE)
                          ;=============================================
                          ;
 1D00                             CSEG 1D00H
                                  ORG 0
                          ;
0000 8B4F0A               FORM13: MOV CX,10[BX]
0003 8EC1                         MOV ES,CX
0005 8B7708                       MOV SI,8[BX]
0008 268A24                       MOV AH,ES:[SI]
                          ;
000B 8B4F06                       MOV CX,6[BX]
000E 8EC1                         MOV ES,CX
0010 8B7704                       MOV SI,4[BX]
0013 268A34                       MOV DH,ES:[SI]
                          ;
0016 8B4F02                       MOV CX,2[BX]
0019 8EC1                         MOV ES,CX
001B 8B37                         MOV SI,0[BX]
001D 268A14                       MOV DL,ES:[SI]
                          ;
0020 53                           PUSH BX          ; SAVE BX TO RETURN PARA
0021 8ADC                         MOV BL,AH
0023 0E                           PUSH CS
0024 07                           POP ES           ; ES<=CS
0025 BD6800                       MOV BP,OFFSET DTBUF
                          ; BL=CYLINDER,DH=SURFACE,DL=SECTOR LENGTH
0028 B91A00               TABLM:  MOV CX,26
002B 8BF5                         MOV SI,BP
                          ;
002D 26881C               TABLM1: MOV ES:[SI],BL
0030 26887401                     MOV ES:1[SI],DH
0034 26885403                     MOV ES:3[SI],DL
0038 83C604                       ADD SI,4
003B E0F0           002D          LOOPNZ TABLM1
                          ;
003D 8AC2                         MOV AL,DL
003F 0AC0                         OR AL,AL
0041 7505           0048          JNE TYPE_D
0043 B8911D                       MOV AX,1D91H     ; SINGLE TYPE,DRIVE 2
0046 EB03           004B          JMPS BIOSUB
0048 B8915D               TYPE_D: MOV AX,5D91H     ; DOUBLE TYPE,DRIVE 2
                          ;
004B 8ACB                 BIOSUB: MOV CL,BL
004D 8AEA                         MOV CH,DL
004F BB6800                       MOV BX,104
0052 B240                         MOV DL,40H       ; 40H='@'
                          ;
0054 CD1B                         INT 1BH
                          ;
0056 5B                           POP BX           ; RESTORE BX
0057 7203           005C          JC ERROR
0059 B80000                       MOV AX,0
005C 8B4F0E               ERROR:  MOV CX,14[BX]
005F 8EC1                         MOV ES,CX
```

```
0061 8B770C                       MOV SI,12[BX]
0064 268904                       MOV ES:[SI],AX
                        ;
0067 CF                 BACK:     IRET                   ; BACK TO BASIC
                        ;
0068 00000100           DTBUF     DB 0,0,01H,0
006C 00000E00                     DB 0,0,0EH,0
0070 00000200                     DB 0,0,02H,0
0074 00000F00                     DB 0,0,0FH,0
0078 00000300                     DB 0,0,03H,0
007C 00001000                     DB 0,0,10H,0
0080 00000400                     DB 0,0,04H,0
0084 00001100                     DB 0,0,11H,0
0088 00000500                     DB 0,0,05H,0
008C 00001200                     DB 0,0,12H,0
0090 00000600                     DB 0,0,06H,0
0094 00001300                     DB 0,0,13H,0
0098 00000700                     DB 0,0,07H,0
009C 00001400                     DB 0,0,14H,0
00A0 00000800                     DB 0,0,08H,0
00A4 00001500                     DB 0,0,15H,0
00A8 00000900                     DB 0,0,09H,0
00AC 00001600                     DB 0,0,16H,0
00B0 00000A00                     DB 0,0,0AH,0
00B4 00001700                     DB 0,0,17H,0
00B8 00000B00                     DB 0,0,0BH,0
00BC 00001800                     DB 0,0,18H,0
00C0 00000C00                     DB 0,0,0CH,0
00C4 00001900                     DB 0,0,19H,0
00C8 00000D00                     DB 0,0,0DH,0
00CC 00001A00                     DB 0,0,1AH,0
                        ;
                                  END
```

5. 8インチIDリーダー
①セクタリード

```
                    ;*************************************
                    ;* READ ID for 8" FD                 *
                    ;* CALLING SEQUENCE                  *
                    ;*     CLEAR ,&H1A00:DEF SEG=&H1A00  *
                    ;*     CC=cylinder address           *
                    ;*     H =head address               *
                    ;*     READID=0:CALL READID(CC,H)    *
                    ;* RETURN INFORMATION                *
                    ;*     from &H1A10 (AX,BX,CX,DX,...) *
                    ;*     BIOS RETURN CODE              *
                    ;*         AH : STATUS               *
                    ;*         CH : SL(0,1,2,3)          *
                    ;*         CL : CC                   *
                    ;*         DH : H                    *
                    ;*         DL : RR                   *
                    ;*************************************
                    ;
                    READID:
0000 C47704               LES     SI,04[BX]          ;
0003 268A0C               MOV     CL,ES:[SI]         ; get cylinder adr.
0006 C437                 LES     SI,00[BX]          ;
0008 268A34               MOV     DH,ES:[SI]         ; get head adr.
000B 8CC8                 MOV     AX,CS              ;
000D 8ED8                 MOV     DS,AX              ; set data segment adr.
000F B8915A               MOV     AX,5A91H           ; set BIOS ID
                                                     ;   & device/unit adr.

0012 33DB                   XOR    BX,BX             ; clear B Reg.
0014 32ED                   XOR    CH,CH             ;       C Reg.
0016 32D2                   XOR    DL,DL             ;       D Reg.
                        RTY:
0018 89870001               MOV    0100H[BX],AX      ; save  A Reg.
001C 899F0201               MOV    0102H[BX],BX      ;       B Reg.
0020 898F0401               MOV    0104H[BX],CX      ;       C Reg.
0024 89970601               MOV    0106H[BX],DX      ;       D Reg.
0028 CD1B                   INT    1BH               ; call  BIOS
002A 7336           0062    JNB    OK1               ; branch if normal

002C F6C4E0                 TEST   AH,0E0H           ;
002F 751B           004C    JNZ    RTY1              ; branch if Missing Data

0031 F6C4C0                 TEST   AH,0C0H           ;
0034 7516           004C    JNZ    RTY2              ; branch if No Data
0036 F6C460                 TEST   AH,060H           ;
0039 7500           003B    JNZ    RET1              ; branch if Not Ready
                        ;   TEST   AH,040H           ;
                        ;   JNZ    RET1              ; branch if Equip. check
003B 89870001           RET1: MOV  0100H[BX],AX      ;
003F 899F0201               MOV    0102H[BX],BX      ;
0043 898F0401               MOV    0104H[BX],CX      ;
0047 89970601               MOV    0106H[BX],DX      ;
004B CF                     IRET                     ; return to BASIC
```

```
                    RTY1:
                    RTY2:                                   ; execute when normal

004C 86A70101               XCHG    AH,0101H[BX]    ;   recover AH


0050 878F0401               XCHG    0104H[BX],CX    ;               CX
0054 87970601               XCHG    0106H[BX],DX    ;               DX
0058 80FC1A                 CMP     AH,1AH          ;
005B 74DE        003B       JE      RET1            ;   return if double

005D B41A                   MOV     AH,1AH          ;   change double

005F E9B6FF      0018       JMP     RTY             ;   and retry it
                     ;
                     OK1:                           ; execute when normal

0062 8AA70101               MOV     AH,0101H[BX]    ;   set single density

0066 878F0401               XCHG    0104H[BX],CX    ;   recover CX
006A 87970601               XCHG    0106H[BX],DX    ;           DX
006E 83C308                 ADD     BX,08H          ;   add entry size
0071 81FBD000               CMP     BX,208          ;
0075 7DC4        003B       JGE     RET1            ;   branch if over
0077 E99EFF      0018       JMP     RTY             ;   try next sector
                            END
```

8インチIDリーダー
②トラックリード

```
                        ;****************************************
                        ;* READ ID for 8" FD                    *
                        ;* CALLING SEQUENCE                     *
                        ;*      CLEAR ,&H1A00:DEF SEG=&H1A00    *
                        ;*                                      *
                        ;*                                      *
                        ;*      READID=0:CALL READID            *
                        ;* RETURN INFORMATION                   *
                        ;*      from &H1A10 (AX,BX,CX,DX,....)  *
                        ;*      BIOS RETURN CODE                *
                        ;*          AH : STATUS                 *
                        ;*          CH : SL(0,1,2,3)            *
                        ;*          CL : CC                     *
                        ;*          DH : H                      *
                        ;*          DL : RR                     *
                        ;****************************************
                        ;
0000 8CC8               READID: MOV   AX,CS       ;
0002 8ED8                       MOV   DS,AX       ; set data segment adr.

0004 B8915A                     MOV   AX,5A91H    ; set BIOS ID

                                                  ;   & device/unit adr.
0007 33DB                       XOR   BX,BX       ; clear B Reg.
```

```
0009 33C9                   XOR   CX,CX          ;         C Reg.
000B 33D2                   XOR   DX,DX          ;         D Reg.
                    RTY:
000D 89870001               MOV   0100H[BX],AX   ; save    A Reg.
0011 899F0201               MOV   0102H[BX],BX   ;         B Reg.
0015 898F0401               MOV   0104H[BX],CX   ;         C Reg.
0019 89970601               MOV   0106H[BX],DX   ;         D Reg.
001D CD1B                   INT   1BH            ; call    BIOS
001F 7336        0057       JNB   OK1            ; branch if normal

0021 F6C4E0                 TEST  AH,0E0H        ;
0024 751B        0041       JNZ   RTY1           ; branch if Missing
                                                 ;
                                                           Address mark
0026 F6C4C0                 TEST  AH,0C0H        ;
0029 7516        0041       JNZ   RTY2           ; branch if No Data
002B F6C460                 TEST  AH,060H        ;
002E 7500        0030       JNZ   RET1           ; branch if Not Ready
                    ;       TEST  AH,040H        ;
                    ;       JNZ   RET1           ; branch if Equip. Check
0030 89870001       RET1:   MOV   0100H[BX],AX   ;
0034 899F0201               MOV   0102H[BX],BX   ;
0038 898F0401               MOV   0104H[BX],CX   ;
003C 89970601               MOV   0106H[BX],DX   ;
0040 CF                     IRET                 ; return to BASIC

                    RTY1:
                    RTY2:                        ; execute when MA or ND
0041 86A70101               XCHG  AH,0101H[BX]   ;   recover AH
0045 878F0401               XCHG  0104H[BX],CX   ;           CX
0049 87970601               XCHG  0106H[BX],DX   ;           DX
004D 80FC1A                 CMP   AH,1AH         ;
0050 74DE        0030       JE    RET1           ;   return if double
0052 B41A                   MOV   AH,1AH         ;   chenge double

0054 E9B6FF      000D       JMP   RTY            ;   and retry it
                    ;
                    OK1:                         ; execute when normal

0057 B45A                   MOV   AH,5AH         ;   set single density

0059 878F0401               XCHG  0104H[BX],CX   ;   recover CX
005D 87970601               XCHG  0106H[BX],DX   ;           DX
0061 83C308                 ADD   BX,08H         ;   BX = BX + 8
0064 80C601                 ADD   DH,01H         ;   count up Head adr.

0067 80FE01                 CMP   DH,01H         ;
006A 74A1        000D       JE    RTY            ;   branch if overflow
006C 32F6                   XOR   DH,DH          ;   set Head address = 0
006E 80C101                 ADD   CL,01H         ;   add 1 to cylinder

0071 80F94D                 CMP   CL,4DH         ;
0074 7DBA        0030       JGE   RET1           ;   branch if over

0076 E994FF      000D       JMP   RTY            ;   try next truck
                            END
```

6．テキスト画面コピー

```
                        ;************************
                        ;* TEXT COPY ROUTINE
                        ;*  CALL TXT(S%,E%)
                        ;************************
                        ;
0000 C437               GETP:   LES SI,00H[BX]
0002 268B0C                     MOV CX,ES:[SI]    ; CX=E%
                        ;
0005 C47704                     LES SI,04H[BX]
0008 268B1C                     MOV BX,ES:[SI]    ; BX=S%
                        ;
000B B800A0             PREP:   MOV AX,0A000H     ; TEXT SEGMENT
000E 8ED8                       MOV DS,AX
0010 53                 OUTLP:  PUSH BX           ; SAVE START
0011 33F6                       XOR SI,SI         ; SI=0 ONE LINE (0-159)
0013 8BC3                       MOV AX,BX
0015 BAA000                     MOV DX,160        ; 160 BYTES SKIP
0018 F7E2                       MUL DX            ; AX=AX*160
001A 8BD8                       MOV BX,AX
                        ;
001C 8D38               INLP:   LEA DI,[BX+SI]    ; DI=EFFECTIVE ADD
001E 8A05                       MOV AL,[DI]
0020 3C1F                       CMP AL,31
0022 7702          0026         JA NEXT
0024 7A04          002A         JP MVAL
0026 3CF8               NEXT:   CMP AL,248
0028 7202          002C         JB LPT            ; LPRINT CHR$(AL)
                        ;
002A B020               MVAL:   MOV AL,' '        ; LPRINT SPACE
002C E80F00        003E LPT:    CALL LPRINT
002F 46                         INC SI
0030 46                         INC SI
0031 81FEA000                   CMP SI,160        ; END ?
0035 75E5          001C         JNE INLP
                        ;
0037 5B                         POP BX            ; RESTORE LINE CONTER
0038 43                         INC BX
0039 3BD9                       CMP BX,CX         ; END OF LINE ?
003B 75D3          0010         JNE OUTLP
                        ;
003D CF                         IRET              ; RETURN TO BASIC
                        ;
003E 56                 LPRINT: PUSH SI
003F B411                       MOV AH,11H        ; LPRINT SUB
0041 CD1A                       INT 1AH
0043 5E                         POP SI
0044 C3                         RET               ; RETURN TO MAIN
                        ;
                                END
```

7．グラフィック画面コピー

①カラー対応画面コピー（640×200モード）

```
                                ;*****************
                                ;* COPY GRAPHICS *
                                ;* SC200.A86     *
                                ;* 640 X 200 MODE*
                                ;*****************
                                ;
0000 1E                         START:  PUSH DS             ; SEG REG INIT
0001 0E                                 PUSH CS
0002 0E                                 PUSH CS
0003 1F                                 POP DS
0004 07                                 POP ES
                                ;
0005 BBB000                     PINIT:  MOV BX,ES:OFFSET PRINTI
0008 B90A00                             MOV CX,0AH
000B E88F00          009D               CALL LPRINT         ; ESC;"M",">","T16"
                                ;
000E 8C16C600                   SSAVE:  MOV SSS,SS          ; STACK SAVE
0012 8926C800                           MOV SPS,SP
0016 8CD8                               MOV AX,DS
0018 8ED0                               MOV SS,AX           ; SET USER STACK
001A BC4A01                             MOV SP,OFFSET STACK
001D CDA0                               INT 0A0H            ; G-LIO INIT
                                ;
001F 33D2                               XOR DX,DX           ; X LOOP COUNTER 0
0021 BBBA00                     XLOOP:  MOV BX,ES:OFFSET BIT
0024 B90600                             MOV CX,06H
0027 E87300          009D               CALL LPRINT         ; ESC;"S0800"
                                ;
002A BBC700                             MOV BX,199          ; Y LOOP 0 - 199
002D BE6001                     YLOOP:  MOV SI,OFFSET DATA
0030 33C0                               XOR AX,AX
0032 B90200                             MOV CX,02H          ; LOOP 2 TIMES
0035 8904                       CLEAR:  MOV [SI],AX         ; CLEAR PRINT BUF
0037 46                                 INC SI
0038 46                                 INC SI
0039 E2FA            0035               LOOP CLEAR
003B BD0100                             MOV BP,1            ; BIT ADD
                                ;
003E B90800                     ADD:    MOV CX,08H          ; LOOP 8 TIMES
0041 33FF                               XOR DI,DI           ; DI=0
                                ;
0043 891EC400                           MOV PARAY,BX        ; PARA Y=BX
0047 53                                 PUSH BX             ; Y=BX SAVE
0048 52                                 PUSH DX             ; X=DX SAVE
0049 B80800                             MOV AX,8            ; X=X*8
004C F7E2                               MUL DX              ; AX=DX:AX*DX
004E 8BD8                               MOV BX,AX
0050 8D01                       LP:     LEA AX,[BX+DI]      ; AX=X=BX+DI=X*8+DI
0052 A3C200                             MOV PARAX,AX        ; STORE X PARA
0055 E84A00          00A2               CALL LIO            ; G-LIO POINT
0058 D0F8                               SAR AL,1            ; AL=COLOR CODE 0-7
005A 3C03                               CMP AL,3            ; AL=AL/2  CODE 0-3
005C 7410            006E               JE L5200
005E 3C00                               CMP AL,0
0060 7502            0064               JNE NEXT
0062 B003                               MOV AL,3
```

```
0064 BE6001              NEXT:   MOV SI,OFFSET DATA
0067 012C                DADD:   ADD [SI],BP        ; STORE BIT IMAGE DATA
0069 46                          INC SI
006A FEC8                        DEC AL
006C 75F9          0067          JNZ DADD
006E 03ED                L5200:  ADD BP,BP          ; BIT IMAGE ADD
0070 47                          INC DI
0071 E2DD          0050          LOOP LP
                         ;
0073 BB6001                      MOV BX,OFFSET DATA
0076 B90400                      MOV CX,04H         ; LPRINT B-IMAGE DATA
0079 E82100        009D          CALL LPRINT
                         ;
007C 5A                          POP DX             ; RESTORE X
007D 5B                          POP BX             ; RESTORE Y
007E 4B                          DEC BX             ; Y=Y-1
007F 83FBFF                      CMP BX,0FFFFH      ; Y LOOP UNTIL BX=0
0082 75A9          002D          JNE YLOOP
                         ;
0084 BBC000              PCRLF:  MOV BX,OFFSET CRLF
0087 B90200                      MOV CX,02H         ; LPRINT 2 CHARA
008A E81000        009D          CALL LPRINT
                         ;
008D 42                          INC DX             ; X=X+1
008E 83FA50                      CMP DX,80          ; IF X=80 THEN RES
0091 758E          0021          JNE XLOOP
                         ;
0093 8E16C600            RES:    MOV SS,SSS         ; RESTORE SS AND SP
0097 8B26C800                    MOV SP,SPS
009B 1F                          POP DS             ; DATA SEG
009C CF                          IRET               ; BACK TO BASIC

                         ;===========================================
                         ;  SUBROUTINES FOLLOW ...
                         ;===========================================
                         ;
009D B430                LPRINT: MOV AH,30H
009F CD1A                        INT 1AH
00A1 C3                          RET                ; RETURN TO MAIN
                         ;
                         ;----------------------------------------
                         ; GRAPHIC LIO POINT
                         ;----------------------------------------
                         ;
00A2 53                  LIO:    PUSH BX            ; SAVE MAIN REGS
00A3 51                          PUSH CX
00A4 57                          PUSH DI
00A5 55                          PUSH BP
                         ;
00A6 BBC200              POINT:  MOV BX,OFFSET PARAX
00A9 CDAF                        INT 0AFH           ; AL=COLOR CODE 0-7

00AB 5D                          POP BP             ; RESTORE COUNTERS
00AC 5F                          POP DI
00AD 59                          POP CX
00AE 5B                          POP BX
00AF C3                          RET                ; RETURN TO MAIN
```

```
                        ;
                        ;****************************
                        ; EXTRA SEGMENT
                        ;****************************
                        ;
 00B0                   DA      EQU OFFSET $
                                ESEG
                                ORG DA
                        ;
00B0 1B4D1B3E1B54       PRINTI  DB 1BH,'M',1BH,'>',1BH,'T16',0DH,0AH
     31360D0A
                        ;
00BA 1B5330383030       BIT     DB 1BH,'S0800'
00C0 0D0A               CRLF    DB 0DH,0AH
                        ;
                        ;++++++++++++++++++++++++++++
                        ; DATA SEGMENT
                        ;++++++++++++++++++++++++++++
                        ;
 00C2                   DATAS   EQU OFFSET $
                                DSEG
                                ORG DATAS
                        ;
00C2                    PARAX   RW 1            ; STORE X
00C4                    PARAY   RW 1            ; STORE Y
                        ;
00C6                    SSS     RW 1            ; STORE SS
00C8                    SPS     RW 1            ; STORE SP
                        ;
00CA                            RB 128          ; STACK FOR LIO
014A                    STACK   RB 16H          ; USER STACK
                        ;
0160                    DATA    RW 1            ; STORE BIT IMAGE DATA
0162                            RW 1
                        ;
                        ;
                                ORG 620H        ; LIO WORK AREA
0620                            RB 0DE0H
                        ;
                                END
```

グラフィック画面コピー
②カラー対応画面コピー（640×400モード）

```
                            ;*****************
                            ;* COPY GRAPHICS *
                            ;* SC400.A86     *
                            ;*640 x 400 MODE *
                            ;*****************
                            ;
0000 1E                     START:  PUSH DS             ; SEG REG INIT
0001 0E                             PUSH CS
0002 0E                             PUSH CS
0003 1F                             POP DS
0004 07                             POP ES
                            ;
0005 BBB500                 PINIT:  MOV BX,ES:OFFSET PRINTI
0008 B90F00                         MOV CX,0FH
000B E89400         00A2            CALL LPRINT         ; ESC;'Q','M','>','T16'
                            ;
000E 8C16D400               SSAVE:  MOV SSS,SS          ; STACK SAVE
0012 8926D600                       MOV SPS,SP
0016 8CD8                           MOV AX,DS
0018 8ED0                           MOV SS,AX           ; SET USER STACK
001A BC5801                         MOV SP,OFFSET STACK
001D CDA0                           INT 0A0H            ; G-LIO INIT
001F BBCC00                         MOV BX,OFFSET SCREEN
0022 CDA1                           INT 0A1H
                            ;
0024 33D2                           XOR DX,DX           ; X LOOP COUNTER 0
0026 BBC400                 XLOOP:  MOV BX,ES:OFFSET BIT
0029 B90600                         MOV CX,06H
002C E87300         00A2            CALL LPRINT         ; ESC;'S1600'
                            ;
002F BB8F01                         MOV BX,399          ; Y LOOP 0 - 399
0032 BE6E01                 YLOOP:  MOV SI,OFFSET DATA
0035 33C0                           XOR AX,AX
0037 B90200                         MOV CX,2            ; LOOP 2 TIMES
003A 8904                   CLEAR:  MOV [SI],AX         ; CLEAR PRINT BUF
003C 46                             INC SI
003D 46                             INC SI
003E E2FA           003A            LOOP CLEAR
0040 BD0100                         MOV BP,1            ; BIT ADD
                            ;
0043 B90800                 ADD:    MOV CX,08H          ; LOOP 8 TIMES
0046 33FF                           XOR DI,DI           ; DI=0
                            ;
0048 891ED200                       MOV PARAY,BX        ; PARA Y=BX
004C 53                             PUSH BX             ; Y=BX SAVE
004D 52                             PUSH DX             ; X=DX SAVE
004E B80800                         MOV AX,8            ; X=X*8
0051 F7E2                           MUL DX              ; AX=DX:AX*DX
0053 8BD8                           MOV BX,AX
0055 8D01                   LP:     LEA AX,[BX+DI]      ; AX=X=BX+DI=X*8+DI
0057 A3D000                         MOV PARAX,AX        ; STORE X PARA
005A E84A00         00A7            CALL LIO            ; G-LIO POINT
005D D0F8                           SAR AL,1            ; AL=COLOR CODE/2
005F 3C03                           CMP AL,3
0061 7410           0073            JE L5200
0063 3C00                           CMP AL,0
```

```
0065 7502       0069          JNE NEXT
0067 B003                     MOV AL,3
0069 BE6E01            NEXT:  MOV SI,OFFSET DATA
                       ;
006C 012C              DADD:  ADD [SI],BP         ; STORE BIT IMAGE DATA
006E 46                       INC SI
006F FEC8                     DEC AL
0071 75F9       006C          JNZ DADD
0073 03ED              L5200: ADD BP,BP
0075 47                       INC DI
0076 E2DD       0055          LOOP LP
                       ;
0078 BB6E01                   MOV BX,OFFSET DATA
007B B90400                   MOV CX,04H          ; LPRINT B-IMAGE DATA
007E E82100     00A2          CALL LPRINT
                       ;
0081 5A                       POP DX              ; RESTORE X
0082 5B                       POP BX              ; RESTORE Y
0083 4B                       DEC BX              ; Y=Y-1
0084 83FBFF                   CMP BX,0FFFFH
0087 75A9       0032          JNE YLOOP           ; REPEAT UNTIL Y=0
                       ;
0089 BBCA00            PCRLF: MOV BX,OFFSET CRLF
008C B90200                   MOV CX,02H          ; LPRINT 2 CHARA
008F E81000     00A2          CALL LPRINT
                       ;
0092 42                       INC DX              ; X=X+1
0093 83FA50                   CMP DX,80           ; IF X=80 THEN RES
0096 758E       0026          JNE XLOOP
                       ;
0098 8E16D400          RES:   MOV SS,SSS          ; RESTORE SS AND SP
009C 8B26D600                 MOV SP,SPS
00A0 1F                       POP DS              ; DATA SEG
00A1 CF                       IRET                ; BACK TO BASIC

                       ;===========================================
                       ; SUBROUTINES FOLLOW ...
                       ;===========================================
                       ;
00A2 B430              LPRINT: MOV AH,30H
00A4 CD1A                     INT 1AH
00A6 C3                       RET                 ; RETURN TO MAIN
                       ;
                       ;-------------------------------------------
                       ; GRAPHIC LIO POINT
                       ;-------------------------------------------
                       ;
00A7 53                LIO:   PUSH BX             ; SAVE MAIN REGS
00A8 51                       PUSH CX
00A9 57                       PUSH DI
00AA 55                       PUSH BP
                       ;
00AB BBD000            POINT: MOV BX,OFFSET PARAX
00AE CDAF                     INT 0AFH            ; AL=COLOR CODE 0-7

00B0 5D                       POP BP              ; RESTORE COUNTERS
00B1 5F                       POP DI
00B2 59                       POP CX
00B3 5B                       POP BX
00B4 C3                       RET                 ; RETURN TO MAIN
```

```
                    ;
                    ;
                    ;*****************************
                    ; EXTRA SEGMENT
                    ;*****************************
                    ;
00B5                DA      EQU OFFSET $
                            ESEG
                            ORG DA
                    ;
                    ;+++++++++++++++++++++++++++++
                    ; DATA SEGMENT
                    ;+++++++++++++++++++++++++++++
                    ;
00CC                DATAS   EQU OFFSET $
                            DSEG
                            ORG DATAS
                    ;
00CC 03000001       SCREEN  DB 03H,00H,00H,01H      ; SCREEN 3,0,0,1
                    ;
00D0                PARAX   RW 1                ; STORE X
00D2                PARAY   RW 1                ; STORE Y
                    ;
00D4                SSS     RW 1                ; STORE SS
00D6                SPS     RW 1                ; STORE SP
                    ;
00D8                        RB 128              ; STACK FOR LIO
0158                STACK   RB 16H              ; USER STACK
                    ;
016E                DATA    RW 1                ; STORE BIT IMAGE DATA
                    ;
                    ;
                    ;
00B5 1B510D         PRINTI  DB 1BH,'Q',0DH
00B8 1B4D0D                 DB 1BH,'M',0DH
00BB 1B3E0D                 DB 1BH,'>',0DH
00BE 1B5431360D0A           DB 1BH,'T16',0DH,0AH
                    ;
00C4 1B5331363030   BIT     DB 1BH,'S1600'
00CA 0D0A           CRLF    DB 0DH,0AH
                            ORG 620H            ; LIO WORK AREA
0620                        RB 0DE0H
                    ;
                            END
```

8．PRINT/LPRINT (CALL文のルーチン)

```
                                  ;==============================
                                  ; PRINT/LPRINT FOR PC-9801
                                  ; Calling sequence :
                                  ;  Segment:1F00H
                                  ;  Offset :0000H 'PL=0
                                  ;  P%=0:PRINT
                                  ;  P%=1:LPRINT
                                  ;  D$='OUTPUT DATA'
                                  ;  CALL PL(P%,D$)
                                  ;==============================
                                  ;
0000 C437                         PARA2:  LES SI,[BX]       ; ES:SI=PARA2
0002 268B0C                               MOV CX,ES:[SI]    ; CL=LEN(D$)
                                                            ; CH=RELOCATION CODE
0005 80FD00                               CMP CH,0          ; WHICH SEGMENT
0008 7405            000F                 JE SEGDX          ; IF CH=0 THEN DX
000A 32ED                                 XOR CH,CH         ; CH=0:CX=LEN(D$)
000C BA6000                               MOV DX,60H        ; IF CH<>0 THEN 60H
                                  ;
000F C47F04                       SEGDX:  LES DI,04[BX]     ; ES:DI=PARA1
0012 268B05                               MOV AX,ES:[DI]    ; AX=P%
                                  ;
0015 268B7402                             MOV SI,ES:02[SI]; SI=D$ OFFSET
0019 8EDA                                 MOV DS,DX         ; DS=D$ SEGMENT
                                  ;
001B 3D0000                       PL:     CMP AX,0          ; IF AX=0 THEN PRINT
001E 7406            0026                 JE PRINT
0020 3D0100                               CMP AX,1          ; IF AX=1 THEN LPRINT
0023 741A            003F                 JE LPRINT
0025 CF                                   IRET              ; IF AX<>1 OR 0 THEN BACK
                                  ;
0026 B86000                       PRINT:  MOV AX,60H        ; AX=60H
0029 8EC0                                 MOV ES,AX         ; ES=60H
002B 51                                   PUSH CX           ; SAVE CX=LEN(D$)
002C BF0202                               MOV DI,202H       ; DEST=PRINT BUFFER
002F FC                                   CLD               ; INC MODE
0030 F3A5                                 REP MOVSW         ; D$ BLOCK TRANSFER
0032 59                                   POP CX            ; RESTORE CX
0033 8ED8                                 MOV DS,AX         ; DS=60H
0035 CDC2                                 INT 0C2H          ; PREPARATION
0037 B800A0                               MOV AX,0A000H     ; AX=TEXT VRAM
003A 8EC0                                 MOV ES,AX         ; ES=TEXT VRAM
003C CD89                                 INT 89H           ; PRINT CALL
003E CF                                   IRET              ; BACK TO BASIC
                                  ;
003F 1E                           LPRINT: PUSH DS
0040 07                                   POP ES            ; ES=DS
0041 56                                   PUSH SI
0042 5B                                   POP BX            ; BX=SI
0043 B86000                               MOV AX,60H
0046 8ED8                                 MOV DS,AX         ; DS=60H
0048 B430                                 MOV AH,30H


004A CD1A                                 INT 1AH           ; LPRINT CALL
004C CF                                   IRET              ; BACK TO BASIC
                                  ;
                                          END
```

9. PRINT/LPRINT (CMD ON/OFF)

```
                               ;=========================
                               ;      PRINT/LPRINT
                               ; with CMD command
                               ; CMD ON : PRINT -> LPRINT
                               ; CMD OFF: Cancel
                               ;=========================
                               ;
 00E8                          CMD     EQU 0E8H
 00BD                          ON      EQU 0BDH
 00BF                          OFF     EQU 0BFH
 00C0                          PRINT   EQU 0C0H
 00AD                          LPRINT  EQU 0ADH
 06EA                          EXEC    EQU 06EAH
 06E8                          NEXT    EQU 06E8H
 159C                          ADDR    EQU 159CH
                               ;
0000 3CE8                      CHECK:  CMP AL,CMD        ; If not CMD then END
0002 F8                                CLC
0003 7539         003E                 JNE PBACK
0005 E80D00       0015                 CALL TOKEN        ; BL=Next Token
0008 80FBBD                            CMP BL,ON
000B 7418         0025                 JE PON            ; PRINT -> LPRINT
000D 80FBBF                            CMP BL,OFF
0010 742C         003E                 JE PBACK
0012 E90A00       001F                 JMP ERROR
                               ;
0015 8936EA06                  TOKEN:  MOV .EXEC,SI      ; Get Token
0019 BF0D00                            MOV DI,13
001C CDC4                              INT 0C4H
001E C3                                RET
                               ;
001F BF0100                    ERROR:  MOV DI,1          ; 'Syntax error'
0022 CDC4                              INT 0C4H
0024 CB                                RETF
                               ;
0025 BB9C15                    PON:    MOV BX,ADDR       ; CMD ON
0028 B84000                            MOV AX,OFFSET AD+1
002B 8907                              MOV [BX],AX
002D E90700       0037                 JMP PEND
                               ;
0030 BB9C15                    POFF:   MOV BX,ADDR       ; CMD OFF
0033 33C0                              XOR AX,AX
0035 8907                              MOV [BX],AX
                               ;
0037 A1EA06                    PEND:   MOV AX,.EXEC      ; Set text pointer
003A A3E806                            MOV .NEXT,AX
003D F9                                STC
003E CB                        PBACK:  RETF
                               ;
                               ;==========================================
                               ;
  003F                         CMDOFF  EQU OFFSET $
                               ;
                                       ORG CMDOFF
                               ;
003F 90                        AD      DB 90H            ; NOP
0040 3CC0                      PL:     CMP AL,PRINT      ; If PRINT then LPRINT
```

```
0042 7407         004B          JE PRNL
0044 3CE8                       CMP AL,CMD          ; If CMD then CMD Process
0046 7407         004F          JE CMDP
0048 E90200       004D          JMP BACK
                      ;
004B B0AD             PRNL:     MOV AL,LPRINT
004D F8               BACK:     CLC
004E CB                         RETF
                      ;
004F E8C3FF       0015 CMDP:    CALL TOKEN          ; If OFF then POFF
0052 80FBBF                     CMP BL,OFF
0055 74D9         0030          JE POFF
0057 80FBBD                     CMP BL,ON
005A 74F1         004D          JE BACK
005C EBC1         001F SNERR:   JMPS ERROR          ; 'Syntax error'
                      ;
                                END
```

10. 漢字フォントをビットイメージで出力

```
                              ;************************
                              ;* KANJI FONT PRINT
                              ;* IN BIT IMAGE
                              ;************************

0000 C437                     START:  LES SI,[BX]      ; GET PARA FROM BASIC
0002 268B14                           MOV DX,ES:[SI]   ; DX=KANJI CODE(16 DOT)
0005 1E                               PUSH DS          ; SAVE DS FOR INT CALL
0006 0E                               PUSH CS
0007 1F                               POP DS
                              ;
0008 B414                     FONT:   MOV AH,14H       ; BIOS COMMAND
000A 8CDB                             MOV BX,DS        ; SEGMENT
000C B9AA00                           MOV CX,OFFSET BUFF      ; FONT BUUFER
000F CD18                             INT 18H          ; INT CALL
                              ;
0011 BFD200                   GET:    MOV DI,OFFSET PWORK     ; BIT STORE WORK
0014 BEAA00                           MOV SI,OFFSET BUFF
0017 46                               INC SI
0018 46                               INC SI           ; SI=FIRST BUFFER
0019 33DB                             XOR BX,BX        ; BX=OUTER LOOP C=0
001B B101                             MOV CL,1         ; SHIFT COUNTER
001D 56                       OUTLP:  PUSH SI          ; SAVE KANJI FONT ADD
001E 57                               PUSH DI          ; SAVE PWORK ADD
001F 53                               PUSH BX          ; SAVE OUTER C
0020 51                               PUSH CX          ; SAVE CL COUNTER
0021 33C0                             XOR AX,AX        ; CLEAR PWORK
0023 51                               PUSH CX
0024 B90800                           MOV CX,8
0027 8905                     CLR:    MOV [DI],AX
0029 47                               INC DI
002A 47                               INC DI
002B E2FA           0027              LOOP CLR
002D 59                               POP CX
002E BFD200                           MOV DI,OFFSET PWORK

0031 32ED                             XOR CH,CH        ; INNER LOOP COUNTER CH=0
0033 8A34                     INLP:   MOV DH,[SI]      ; DX=FIRST DOT STRING
0035 8A5401                           MOV DL,01H[SI]
0038 D3E2                             SHL DX,CL        ; SHIFT LEFT CL CF=BIT
003A 9F                               LAHF             ; AH=CF
003B F6C401                           TEST AH,1        ; CF=1?
003E 7504           0044              JNE NEXT         ; IF CF=0 THEN AL=0
0040 B000                             MOV AL,0         ; IF CF=1 THEN AL=1
0042 7A02           0046              JP STR
0044 B001                     NEXT:   MOV AL,1
0046 8805                     STR:    MOV [DI],AL      ; STORE
0048 47                               INC DI
0049 83C602                   NEXLP:  ADD SI,02H       ; SKIP 2 BYTES
004C FEC5                             INC CH           ; COUNT UP
004E 80FD10                           CMP CH,16        ; 16 LOOPS?
0051 75E0           0033              JNE INLP         ; INNER LOOP
                              ;
0053 BFD200                   PICKAL: MOV DI,OFFSET PWORK
0056 B90200                           MOV CX,2         ; LOOP 2
0059 E82400         0080      PCHR:   CALL PICKUP
005C 8826D100                         MOV XLOW,AH      ; STORE LOW BYTE
```

```
0060 E81D00      0080          CALL PICKUP
0063 8AC4                      MOV AL,AH
0065 B210                      MOV DL,16           ; YLOW=Y*16
0067 F6E2                      MUL DL
0069 0206D100                  ADD AL,XLOW         ; AL=BYTE TO PRINTER
006D E83100      00A1          CALL PUTC           ; OUTPUT TO PRINTER
0070 E2E7        0059          LOOP PCHR

0072 59                        POP CX              ; RESTORE CL
0073 5B                        POP BX
0074 5F                        POP DI              ; RESTORE KWORD ADD
0075 5E                        POP SI              ; RESTORE KANJI ADD
0076 FEC1                      INC CL              ; SHIFT BIT COUNT UP
0078 43                        INC BX              ; OUTER LOOP COUNT UP
0079 83FB10                    CMP BX,16
007C 759F        001D          JNE OUTLP
                     ;++++++++++++++++++++++++++++++++++++++++
007E 1F                        POP DS
007F CF                        IRET                ; BACK TO BASIC
                     ;++++++++++++++++++++++++++++++++++++++++
                     ;
0080 51              PICKUP:   PUSH CX
0081 32E4                      XOR AH,AH
0083 B101                      MOV CL,1            ; LOOP 4 COUNTER
0085 8A05            AD:       MOV AL,[DI]
0087 0AC0                      OR AL,AL
0089 7403        008E          JZ ZERO
008B E80C00      009A          CALL CALC           ; 1,2,4,8
008E 02E0            ZERO:     ADD AH,AL
0090 47                        INC DI
0091 FEC1                      INC CL
0093 80F905                    CMP CL,5
0096 75ED        0085          JNE AD
0098 59                        POP CX
0099 C3                        RET
                     ;
                     ;------------------------------------------------
                     ;---SUBROUTINE-----------------------------------
                     ;
009A 8AC1            CALC:     MOV AL,CL
009C BBCC00                    MOV BX,OFFSET TBL        ; 1,2,4,8
009F D7                        XLAT AL             ; AL=1,2,4,8
00A0 C3                        RET
                     ;
00A1 1E              PUTC:     PUSH DS
00A2 16                        PUSH SS
00A3 1F                        POP DS
00A4 B411                      MOV AH,11H          ; BIOS CMD
00A6 CD1A                      INT 1AH             ; OUT 1 BYTE (AL)
00A8 1F                        POP DS
00A9 C3                        RET
                     ;
                     ;================================================
                     ; DATA SEGMENT
                     ;================================================
```

```
                              ;
  00AA                        DATA EQU OFFSET $
                              ;
                                      DSEG
                                      ORG DATA
                              ;
00AA                          BUFF    RS 34
00CC 0001020408               TBL     DB 00H,01H,02H,04H,08H
00D1                          XLOW    RS 1
00D2                          PWORK   RS 16
                              ;
                                      END
```

付録－2
ROM内ルーチンのINTによる利用

(1) ＩＮＴ割り込みベクター覧表

ベクタアドレス	ベクタ番号	項　目　名	機　　　能
0～3	0	除算エラー	ROM内のIRET命令をさしている。
4～7	1	シングルステップ	ROM内のIRET命令をさしている。
8～B	2	NMI Non Maskable Interrupt	メモリーエラーチェックルーチンに入っている。
C～F	3	INT 3	ROM内のIRET命令をさしている。
10～13	4	オーバーフロー	ROM内のIRET命令をさしている。
14～17	5	[ＣＯＰＹ]キー	[ＣＯＰＹ]キーをおしたとき飛んでゆく
18～1B	6	[ＳＴＯＰ]キー	[ＳＴＯＰ]キーをおしたとき飛んでゆく
1C～1F	7	インターバルタイマ	ＲＯＭ内のＩＲＥＴ命令をさしている
20～23	8	タイマー	ＲＯＭ内の処理ルーチン
24～27	9	キーボード	キースキャン
28～2B	A	ＣＲＴＶ	
2C～2F	B	拡張バスＩＮＴ０	
30～33	C	ＲＳ-232Ｃ	
34～37	D	拡張バスＩＮＴ1	カセット（ＣＭＴ）
38～3B	E	拡張バスＩＮＴ2	ＯＤＡプリンタ
3C～3F	F	システム予約	
40～43	10	セントロプリンタ	
44～47	11	拡張バスＩＮＴ3	5″ハードディスク
48～4B	12	拡張バスＩＮＴ4	
4C～4F	13	8″フロッピーディスク	
50～53	14	拡張バスＩＮＴ5	
54～57	15	拡張バスＩＮＴ6	
58～5B	16	8087	
5C～5F	17	ノイズ（システム予約）	
60～63	18	キーボード，ＣＲＴ	

ベクタ番号18H

ＩＮＴ　18H

は，キーボード，ＣＲＴ関係のＢＩＯＳです。ＡＨレジスタに，コマンドNo.をセットします。

0－4	キーボード
A－19	テキストＶＲＡＭ
40－4A	グラフィックＶＲＡＭ

コマンドNo. AH	
0	読み出したキーデータコードをＡＸに入れて戻る。 ＡＨ…スキャンコード ＡＬ…内部コード
1	バッファ先頭のキーコードをＡＸに入れて戻る。 ＢＨ…１なら有効，０無効

となっています。40～4AグラフィックVRAMのコマンドは，グラフィック画面の章で説明致しておりますので，そちらをご覧下さい。

コマンドNo. AH	機能
2	AL…ビット状態以下のキーがおされているとき1 0 0 0 [CTRL] [GRAPH] [カナ] [CAPS] [SHIFT]
3	ワークエリア，バッファ，キーボードインターフェースを初期化する。
4	入力 AL…キーコードグループNo.（0～FH） 出力 AH…キーコードグループ内の8つのキーのキーイン状態を8ビットで格納
0AH	CRTモードの設定 AL 0 0 0 0 0 [ ] [ ] [ ] ビット0: 0 25行 / 1 20行 ビット1: 0 80文字モード / 1 40文字モード ビット2: 0 バーティカルライン / 1 簡易グラフ
0BH	CRTモードのセンス AL [ ] 0 0 0 0 [ ] [ ] [ ] ビット0～2: 上と同じ ビット7: 0 標準ディスプレイ / 1 専用高解像度ディスプレイ

<table>
<tr><th colspan="2">機　　　能</th></tr>
<tr><td>コマンド
No.
AH</td><td></td></tr>
<tr><td>0CH</td><td>テキスト画面表示開始</td></tr>
<tr><td>0DH</td><td>テキスト画面表示停止</td></tr>
<tr><td>0EH</td><td>テキスト画面のＨＯＭＥの位置を変更する。
ＤＸ…テキストＲＡＭ上のオフセットアドレス
（セグメントアドレスはＡ０００Ｈ）</td></tr>
<tr><td>0FH</td><td>テキスト画面に複数の画面を定義する。
ＢＸ…リストのセグメントアドレス
ＣＸ…リストのオフセットアドレス
ＤＨ…リストで最初に定義するエントリの表示領域番号（０～３）
ＤＬ…リストで定義するエントリ個数（１～４）

表示行数
l_0 l_1 l_2 l_3
テキスト画面
テキストVRAM
S_0(オフセットアドレス)
GDCからみたアドレス
⓪ ① ② ③
S_1 S_2 S_3

BX：CX　　リスト
<table><tr><td>S_0</td><td>l_0</td></tr><tr><td>S_1</td><td>l_1</td></tr><tr><td>S_2</td><td>l_2</td></tr><tr><td>S_3</td><td>l_3</td></tr></table>2バイト　2バイト</td></tr>
<tr><td>10H</td><td>ＡＬ＝０　カーソルをブリンクさせる
　　　１　カーソルをブリンクさせない</td></tr>
<tr><td>11H</td><td>上記設定のカーソルを表示</td></tr>
<tr><td>12H</td><td>カーソルを消す</td></tr>
<tr><td>13H</td><td>ＤＸ（ＣＰＵアドレス）に設定したテキストＶＲＡＭアドレスにカーソルを表示する。</td></tr>
</table>

機　　　能	
コマンド No. AH	
14H	フォントパターンをよみ出す。

BX…フォントパターンバッファ先頭セグメントアドレス

CX…フォントパターンバッファ先頭オフセットアドレス

DX…コード

• ANKの場合

DL…コード

DH…{ 00H　標準CRT（6×7フォント 10バイト／バッファ）
80H　専用高解像度CRT（7×13フォント 18バイト／バッファ）

	DH	DL
例：DX	80	2F

7×13フォントで「年（2FH)」のフォントパターンをよみ出す場合

• 漢字の場合

DX…コード

	DH	DL
例：DX	34	41

「漢（3441H)」のフォントパターンをよみ出す場合

例：$\frac{1}{4}$角の場合はANKコードと同じです。

出力形式

①ANK（6×7），漢字$\frac{1}{4}$角

②ANK（7×13）

③漢字全角

	機　　　能
コマンド No. AH	
15H	ライトペンのセンス 出力　AH…｛0ライトペンがおされている 　　　　　　1ライトペンがおされていない DX…ライトペンのおされているテキストVRAMのアドレス（CPU例）
16H	テキストVRAMの全領域をクリア DH…アトリビュートエリアをクリアするデータ DL…表示エリアをクリアするデータ （空白なら20H）
17H	ブザーON
18H	ブザーOFF
19H	ライトペン押下状態の初期化 コマンド15H（ライトペンのセンス）を行なう前に必ず実行すること。
40H～4AH	グラフィック画面に関するものです。グラフィック画面の章で説明しておりますのでそちらをご覧下さい。

ベクタアドレス	ベクタ番号	項　目　名	機　　　能
64～67	19	RS-232C	
INT19HはRS-232C関係のBIOSです。			

コマンド No. AH	
00	RS-232Cの初期化

コマンド No. AH	機　　能
00	R S232Cインターフェースの初期化。

入力　A L…伝送速度

A L	ボーレイト
00	75
01	150
02	300
03	600
04	1200
05	2400
06	4800
07	9600

C H…8251 Aのモード（非同期モード）

bit 0，1 ボーレート　　2…×16，3…×64

bit 2，3 キャラクタ長

0	5ビット
1	6ビット
2	7ビット
3	8ビット

bit 4　　パリティイネーブル（0ディスエイブル，1イネーブル）

bit 5　　0…奇数パリティ，1…偶数パリティ

bit 6，7　ストップビット長

1	1ビット
2	1.5ビット
3	2ビット

C L…8251 Aのコマンド

bit		内　　容
0	TXEN	送信イネーブル　0：ディスエイブル　1：イネーブル
1	DTR	データターミナルReady　0：Busy　1：Ready
2	RXE	受信イネーブル　0：ディスエイブル　1：イネーブル
3	SBRK	Break　0：Break送信しない　1：Break送信する
4	ER	エラーリセット　0：エラーリセットしない　1：エラーリセットする
5	RTS	送信要求　0：OFF　1：ON
6	IR	内部リセット　0：しない　1：する

コマンド No. AH	機　　能
00Hのつづき	ES：DI…受信バッファの先頭アドレス（セグメント：オフセット） DX　　…受信バッファの大きさ（バイト単位） BH　　…送信時タイムアウト時間（×500msec） BL　　…受信時タイムアウト時間（×500msec） 出力　AH…0　正常終了 RS232Cバッファの構成

<table>
<tr><th>コマンド
No.
AH</th><th>機　　能</th></tr>
<tr><td>01H</td><td>フロー制御を行なうRS-232C初期化。
↓
データ受信割り込み時(RXRDY…割り込み番号0CH)に受信バッファの有効データ長がバッファの大きさの$\frac{3}{4}$以上になったとき，送信側にCTRL-Sコード(13H)を出力し，送信を停止することを要求します。
受信データの引き取り(Recieveコマンド)処理時に，受信バッファの有効データ長が，バッファの大きさの$\frac{1}{4}$以下になったとき，送信側に，CTRL-Qコード(11H)を出力し，送信停止状態を解き，送信を再開することを要求します。</td></tr>
<tr><td>02H</td><td>CXに，受信データの長さを得る。
データとステータス(合わせて2バイト)で1データです。</td></tr>
<tr><td>03H</td><td>AL内のデータを送信します。
<table>
<tr><th>AHに
リターン
コード</th><th>内　　容</th></tr>
<tr><td>0</td><td>正常終了</td></tr>
<tr><td>1</td><td>RS−232Cインターフェースの初期化が
行なわれていない。</td></tr>
<tr><td>2</td><td>受信割り込み処理において、
受信バッファがオーバーフローした。</td></tr>
<tr><td>3</td><td>送、受信処理において、8251Aからの
送、受信可のステータスを引き取れなかった。</td></tr>
</table>
</td></tr>
<tr><td>04H</td><td>データ受信。CHにデータ，CLにステータスが入る。</td></tr>
<tr><td>05H</td><td>RS-232CへAL内のコマンドを出力します。</td></tr>
<tr><td>06H</td><td>RS-232Cのステータスを得る。
出力
8251Aステータス
<table>
<tr><th>ビット</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>略称</td><td>DSR</td><td>SYN
DET</td><td>FE</td><td>OE</td><td>PE</td><td>TXE</td><td>RXRDY</td><td>TXRDY</td></tr>
<tr><td>1</td><td>ON</td><td>ブレーク
検出</td><td>エラーあり</td><td>エラーあり</td><td>エラーあり</td><td>バッファ空</td><td>Ready</td><td>Ready</td></tr>
<tr><td>0</td><td>OFF</td><td>検出なし</td><td>エラーなし</td><td>エラーなし</td><td>エラーなし</td><td>バッファ満</td><td>Busy</td><td>Busy</td></tr>
</table>
</td></tr>
</table>

コマンド No. AH	機　　　能
	CL　システムポートステータス bit 5　$\overline{\mathrm{CD}}$　0：受信キャリア検出あり 　　　　　　1：受信キャリア検出なし bit 6　$\overline{\mathrm{CS}}$　0：送信可 　　　　　　1：送信不可

ベクタアドレス	ベクタ番号	項目名	機　　能
68～6B	1A	カセット，プリンタＢＩＯＳ	

ＩＮＴ　１Ａはカセット，プリンタ関係のＢＩＯＳです。

コマンド	
1～5	CMT
10～12H　30H	プリンタ

コマンド No.	機　　　能
AH	セントロニクスパラレルプリンタ
10H	プリンタインターフェース，ステータスエリアの初期化 出力　ＡＨ　bit 0 が { 0…データ送信できない 　　　　　　　　　　 1…データ送信可能
11H	ＡＬ内のデータをプリンタに出力します。 出力　ＡＨ　bit 0 { 0…データ未出力 　　　　　　　　 1…データ出力完了 　　　　　　bit 1 { 0…データ出力完了 　　　　　　　　 1…タイムアウトでデータ出力できなかった。
12H	プリンタのステータスのセンス 出力　ＡＨ　bit 0… { 0…データ出力できない（Busy） 　　　　　　　　　 1…データ出力可能　　（Ready）
30H	複数バイトのデータ出力 入力　ＢＸ…データバッファのオフセットアドレス 　　　ＥＳ…データバッファのセグメントアドレス 　　　ＣＸ…データのバイト数（データバッファはセグメント境界をまたがってはいけない） 出力　ＡＨ…bit 1 { 0…正常終了 　　　　　　　　 1…タイムアウト発生，未出力データがある。 　　　　　　　　 そのときの位置はＢＸにある。（ＣＸ…未出力バイト数）

<table>
<tr><th>コマンド
No.
AH</th><th>機　　　　能</th></tr>
<tr><td></td><td>ODAシリアルプリンタ</td></tr>
<tr><td>10H～12H</td><td>コマンド10H～12H，30Hの動作は同じです。
出力のステータス情報が異なります。
出力　AH
bit0…{0データ送信不可能
　　　{1データ送信可能(RDA)
bit1……1ならタイムアウト
bit4……1ならプリンタの電源がON
bit5……1なら用紙残少又は用紙切れ(Media Lavo, PE)
bit6……1ならプリンタのハードエラー(Alarm又はFault)
bit7……1ならプリンタがデータ受信可能(RMR,セレクト)</td></tr>
<tr><td>30H</td><td>出力　AH
00　正常終了
02　タイムアウト
03　用紙残少又は用紙切れ
62　プリンタ未接続、プリンタ電源OFF
63　通信不可
64　プリンタにハードエラー発生</td></tr>
<tr><td></td><td>カセット</td></tr>
<tr><td>00H</td><td></td></tr>
<tr><td>01H</td><td>カセットのモータOFF。出力AH…正常終了00</td></tr>
<tr><td>02H</td><td>カセットのモータON。データ読み取り可能な状態にする
出力AH…正常終了00
入力AL…{00H…600ボー
　　　　{80H…1200ボー</td></tr>
<tr><td>03H</td><td>カセットのモータON。データ書き込み可能な状態にする。</td></tr>
<tr><td>04H</td><td>データを書き込む。データはALに入れる。(03Hを実行済のこと)
出力AH…0正常終了，2-タイムアウト</td></tr>
<tr><td>05H</td><td>データを読み込む。データはALに入ってくる。(02Hを実行済のこと)
入力　AL…リード時エラー処理{0…エラーを報告せよ。
　　　　　　　　　　　　　　{1…エラーを無視せよ。
出力　AH…0…正常終了
　　　　　2…タイムアウト
　　　　　27H…テープリードエラー（フレーミングエラー，オーバーランエラー）</td></tr>
</table>

ベクタ アドレス	ベクタ 番号	項　目　名
6C～6F	1BH	フロッピーディスク（5インチ，8インチ）のBIOS。 ディスクの章で詳細説明している。
70～73	1CH	カレンダ，インターバルタイマのBIOS。
	コマンド No. AH	機　　能
	00H	ES：BXにセットしたバッファに，日付・時刻をよみ出す 年 10の位 ¦ 年 1の位 ¦ 月 ¦ 曜日 ¦ 日 10の位 ¦ 日 1の位 ¦ 時 10の位 ¦ 時 1の位 ¦ 分 10の位 ¦ 分 1の位 ¦ 秒 10の位 ¦ 秒 1の位 ↑ ES:BX　月と曜日は16進，それ以外はBCD表現で入っています。
	01H	日付・時刻をセットします。バッファはES：BXに設定 し，形式はコマンド00Hと同じです。
	02H	インターバルタイマ割り込みのセット。 CX…インターバルタイマ値（×10mec） 最大655360msec≒11分（CX＝0000H） ES：BX タイムアウトになったときのユーザー処理ルーチン。 例えば，CX＝1にしてこのコマンドを発行すると，10msecごとにINT 7割り込みがかかって，ES：BX処理ルーチンに制御が移ります。 （何もないときINT　7のベクタはROM内のIRETをさしています。） 使用法の一例が335ページの「14－14リアルタイムで時間表示」にあります。

ベクタアドレス	ベクタ番号	項　目　名	機　　能	
74～77	1DH	システム予約	何もないときはROM内のIRET命令をさしている。	
78～7B	1EH	N_{88}-BASIC (86)	コールドスタート	
7C～7F	1FH	システム予約	システム予備の領域D800：2A00をさしている。	
80～FF	20～3F	システム予約	ROM内のIRET命令をさしている。	
100～1FF	40～7F	ユーザー用	ROM内のIRET命令をさしている。ユーザーが定義して自由に使えます。	
200～203	80H	初期化	キーボード，CRT　LIOの初期化	
204～207	81H		処理なし	
208～20B	82H	WIDTH	WIDTH文の下うけ処理をする。33Hのステータスをみながら，ポート60H，62HにデータをOUTする。	
20C～20F	83H	センスInterrupt	キーボードからの割り込み状態をセンスする。	
210～213	84H	INPUT　A$	キーインデータ処理	
214～217	85H	INPUT　WAIT	INPUT　WAIT文の下うけ処理をする。	
218～21B	86H	キーインライン	キーインライン処理をする。	
21C～21F	87H	INPUT$	INPUT$文の下うけ処理をする。	
220～223	88H	INKEY$	INKEY$文の下うけ処理をする。	
224～227	89H	PRINT	PRINT文の下うけ処理をする。	
228～22B	8AH	BEEP	BEEP文の下うけ処理をする。 AL…1　BEEP1	
22C～22F	8BH	スクロール	AL…1　スクロールダウン 0　スクロールアップ	
230～233	8CH	C_R	次の論理行へカーソル設定。	N-BASIC (86) 用
234～237	8DH	P (RE)SET	N_{88}-BASIC(86)では未処理	
238～23B	8EH	POINT	N_{88}-BASIC(86)では未処理	
23C～23F	8FH	GET@	N_{88}-BASIC(86)では未処理	
240～243	90H	GET@A	N_{88}-BASIC(86)では未処理	
244～247	91H	PUT@	N_{88}-BASIC(86)では未処理	
248～24B	92H	PUT@A	N_{88}-BASIC(86)では未処理	
24C～24F	93H	BOX	N_{88}-BASIC(86)では未処理	
250～253	94H	ラインアトリビュートの設定		
254～257	95H	COLOR@	COLOR@(X1, Y1)－(X2, Y2) 文の下うけ処理	

ベクタアドレス	ベクタ番号	項　目　名	機　　能
258～25B	96H	ON TIME GOSUB	ON　TIME　GOSUB文の下うけ処理をする。
25C～25F	97H	KINPUT	日本語入力処理
260～263	98H	CLS	テキスト画面全体のクリアをする。 ES← [1412H] をしておくこと。
264～267	99H	KEY　LIST	文字コードの表示。 アキスコード0DH(キャリッジリターン)等をC_Rなど文字コードで表示する。 KEY　LISTルーチンで使用している。 [使用法] DS←60H ES← [1412,1413H] CX←文字数＊2（バイト） バッファ セグメント60H：オフセット0 2バイトで1文字を表わす。 [436,437H] ←Y座標 [438,439H] ←X座標（これはINT　89H　PRINTにも適用される。）
268～26B	9AH	PEN	ライトペン入力処理
26C～26F	9BH	C_R	次の物理行へカーソルを設定。
270～273	9CH	CTRL-U	物理行をクリアする。 [使用法] DS←60H ES← [1412,1413H] DX←クリアする行 [482H] ←クリアするキャラクタ [47FH] ←クリアするアトリビュートコード
274～277	9DH	ファンクションキー	ファンクションキー行の表示
278～27B	9EH	キーインバッファクリア	LIO／BIOSのキーインバッファをクリアする。
27C～27F	9FH	システム予約	カーソルSWのチェックルーチンにとんでいる。

ベクタアドレス	ベクタ番号	項目名	機能
280～283	A0H	グラフィックＬＩＯ初期化	[使用法]については，グラフィック画面の章で詳細に説明している。（ＩＮＴ Ａ０Ｈ～ ＩＮＴ ＡＦＨ はグラフィック ＬＩＯのコマンドです。）
284～287	A1H	ＳＣＲＥＥＮ	
288～28B	A2H	ＶＩＥＷ	
28C～28F	A3H	ＣＯＬＯＲ１（色処理）	
290～293	A4H	ＣＯＬＯＲ２（パレット処理）	
294～297	A5H	ＣＬＳ（グラフィック画面のＣＬＳ）	
298～29B	A6H	ＰＳＥＴ／ＰＲＥＳＥＴ	
29C～29E	A7H	ＬＩＮＥ	
2A0～2A3	A8H	ＣＩＲＣＬＥ	
2A4～2A7	A9H	ＰＡＩＮＴ１（色でぬりつぶす）	
2A8～2AB	AAH	ＰＡＩＮＴ２（タイルパターンでぬりつぶす）	
2AC～2AF	ABH	ＧＥＴ	
2B0～2B3	ACH	ＰＵＴ１（画）	
2B4～2B7	ADH	ＰＵＴ２（漢字）	
2B8～2BB	AEH	ＲＯＬＬ	
2BC～2BF	AFH	ＰＯＩＮＴ	
2C0～2C3	B0H	ＤＩＳＫ ＬＩＯ	
2C4～2C7	B1H	システム予約	ＲＯＭ内のＩＲＥＴをさしている（ハードディスクＢＩＯＳ用）
2C8～2CB	B2H	システム予約	ＲＯＭ内のＩＲＥＴをさしている
2CC～2CF	B3H	システム予約	ＲＯＭ内のＩＲＥＴをさしている
2D0～2D3	B4H	ＤＩＳＫ ＬＩＯ初期化	
2D4～2FF	B5H～BFH	システム予約	ＲＯＭ内のＩＲＥＴをさしている
300～303	C0H	ＣＯＰＹ	ハードコピー処理
304～307	C1H	コード変換１（キーボード／ＣＲＴ ＬＩＯ）	
308～30B	C2H	コード変換２（キーボード／ＣＲＴ ＬＩＯ）	
30C～30F	C3H	ＣＡＬＬ，ＵＳＲ	N_{88}-ＢＡＳＩＣ（86）からＣＡＬＬ，ＵＳＲでユーザーのマシン語を実行するときに使う。
310～313	C4H	ＲＡＭ部にあるDISK LIOやモニタからＲＯＭ部のサブルーチンを呼び出すのに使用される。（「ＲＯＭ内ルーチンの利用（ＩＮＴ Ｃ４Ｈ）」（421ページ）の章で詳細に説明する。）	

ベクタアドレス	ベクタ番号	項目名	機能
314～317	C5H	グラフィックＬＩＯから割り込みをセンスするエントリー。	
318～31B	C6H	ＤＩＳＫ　ＢＡＳＩＣのスタートエントリー	
31C～31F	C7H	ＤＩＳＫ版Edit機能	
320～323	C8H	N_{88}-ＢＡＳＩＣ(86)	システム予約　ＲＯＭ内のＩＲＥＴをさしている
324～327	C9H	N_{88}-ＢＡＳＩＣ(86)	システム予約　ＲＯＭ内のＩＲＥＴをさしている
329～32B	CAH	N_{88}-ＢＡＳＩＣ(86)	システム予約　ＲＯＭ内のＩＲＥＴをさしている
32C～32F	CBH	ＴＥＲＭ	ＴＥＲＭ文の処理をする。
330～333	CCH	リモートＢＡＳＩＣプロトコルにおけるＢＡＳＩＣステートメント，実行結果の回線への送信処理	
334～337	CDH	リモートＢＡＳＩＣプロトコルの処理	
338～33B	CEH	ＣＯＰＹ	グラフィック画面をＲＡＭ上のバッファによみ出す。
33C～33F	CFH	ＭＯＮ	モニタＲＯＭ部のエントリー
340～343	D0H	ＭＯＮ	モニタＲＡＭ部のエントリー コマンドＡ，Ｌ，^R，^W
344～347	D1H	ＧＰ-ＩＢ	ＧＰ-ＩＢ ＢＩＯＳ起動用
348～34B	D2H	表示選択機能用	
34C～3C3	D3H ～ F0H	N_{88}-ＢＡＳＩＣ(86) システム予約	ＲＯＭ内のＩＲＥＴをさしている
3C4 ～ 3FF	F1H ～ FFH	ユーザー用	ユーザーが自由に定義して使える。ＩＮＴ命令を使えば2バイトでセグメント間コールができる。

(2)ＩＮＴ　Ｃ４Ｈのソフトウエアインターフェイスの説明

ＤＩＳＫ-ＢＡＳＩＣ等ＲＡＭ上のプログラムは，ＩＮＴ　Ｃ４Ｈを用いてテキストの解析，エラーメッセージの表示などを行なっています。
ここでは，その使用法を説明します。
このルーチンを利用するときは，セグメントレジスタ，フラグを以下のように設定する必要があります。

ＤＳ←60Ｈ，ＳＳ←60Ｈ
フラグ　ＩＮＴ　Ｃ４Ｈによって呼び出すときは，関係ない。
（ＩＮＴ命令は，フラグをスタックにPUSHするから。）
Far call CALL Fによって呼び出すときは，
ＣＬＤ　ＤＦ＝０（ディレクションフラグ＝０）増加方向。
ＳＴＩ　ＩＦ＝１（インタラプトフラグ＝１）インタラプトイネーブル。
にすること。

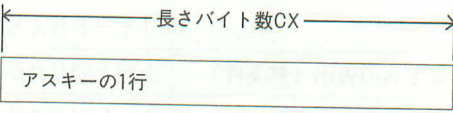
ＤＩにコマンドNo.をセットして，ＩＮＴ　Ｃ４Ｈ（ＣＤ　Ｃ４）を実行します。
コマンドNo.は実に，０～82（10進）の83個もあります。
その一覧表を示します。

ベクタエントリーNo.	機　能	ベクタエントリーNo.	機　能
0	エラーメッセージの表示	19	ファイルディスクリプタのコード化
1	Syntax error表示	20	デバイスタイプコードの取得
2	Illegal function call表示	21	A00H 番地の内容番地をコール
3	Type mismatch表示	22	変数アドレスの取得
4	エラーメッセージ表示後，RETURN	23	ACC-1へのロード
5	倍精度加算	24	ACC-1のストア
6	倍精度乗算	25	ストリング値のロード
7	倍精度除算	26	コンマのスキップ
8	倍精度の整数化（ＩＮＴ）	27	指定行から実行を開始
9	倍精度化（ＣＤＢＬ）	28	次の行を実行
10	オーバフロー処理	29	ＶＡＬ関数と同等
11	ゼロ除算処理	30	ダイレクトモードエントリ
12	数値の整数化（ＣＩＮＴ）	31	プログラム編集時のステータスリセット
13	トークン抽出	32	ＰＲＩＮＴ　ＵＳＩＮＧ用編集
14	数式評価	33	値の編集（STR$）
15	Temporary string descriptorの生成	34	スペースアイテムのスキップ
16	テキスト→ソースイメージ変換	35	文の終端判定
17	行番号のバイナリ化	36	バイナリ→10進数変換（ＳＴＲ$）
18	ファイル番号の評価，チェック	37	カレントデバイスへの出力

ベクタエントリーNo.	機　　能	ベクタエントリーNo.	機　　能
38	ベクタNo.1と同じ	61	1文字CRTへ表示
39	ベクタNo.3と同じ	62	カーソルのリセット
40	ベクタNo.2と同じ	63	C_R, L_Fの出力
41	実数化（CSNG）	64	CRTへC_R, L_F出力
42	倍精度化（CDBL）	65	CRTカーソルのリセット
43	実数加算	66	符号なし整数化
44	実数減算	67	添字評価
45	実数乗算	68	ストリングエリアゴミ集め
46	実数除算	69	符号なし整数の実数化
47	実数比較	70	〝in ×××××〟表示
48	倍精度減算	71	文の読みとばし
49	倍精度比較	72	DATA文の読みとばし
50	バイナリ→8進数変換（OCT$）	73	文字列定数の読みとばし
51	バイナリ→16進数変換（HEX$）	74	数式の評価及び結果の整数化
52	符号なし整数値→10進変換	75	キー，タイマ割り込みのセンス
53	テキストアドレス→行番号書き換え	76	COM，PEN割り込みのセンス
54	テキストから1項目抽出	77	1行トランスレート
55	指定行番号をもつテキストアドレスの取得	78	メモリスイッチの内容取得
56	ベクタNo.55と同じ	79	未実装RAMアクセスチェック
57	テキストのサーチ	80	ストリングエリアの割り付け
58	数式の存在のチェック	81	キーワードのサーチ
59	CRTへの表示	82	キーボードより1行入力
60	CRTへの表示（無条件）		

次に使用法を説明します。

DI	機　　能	使用法・注意点
0	エラーメッセージの表示	表示後ダイレクトモードに入ります。
1	Syntax error表示	表示後ダイレクトモードに入ります。
2	Illegal function call表示	表示後ダイレクトモードに入ります。
3	Type mismatch表示	表示後ダイレクトモードに入ります。
4	エラーメッセージの表示	表示後IRETします。
5	倍精度加算	FAC 1　FAC　　FAC＝FAC 1 ＋FAC 1414 型 141F 1415 符号ビット 1420 1416 1421 1417 1422 1418 1423 1419 1424 141A 1425 141B 1426 141C 1427 141D 整数（1424/141A～1425/141B） 単精度（1424/141A～1427/141D） 倍精度（1420/1416～1427/141D）
6	倍精度乗算	ＦＡＣ＝ＦＡＣ１＊ＦＡＣ
7	倍精度除算	ＦＡＣ＝ＦＡＣ１／ＦＡＣ
8	倍精度の整数化	ＦＡＣ＝ＩＮＴ（ＦＡＣ）
9	倍精度化	ＦＡＣ＝ＣＤＢＬ（ＦＡＣ）
0AH	オーバーフロー処理	ＯＶと表示して戻る。
0BH	ゼロディバイド処理	0/と表示して戻る。
0CH	数値の整数化	ＦＡＣ＝ＣＩＮＴ（ＦＡＣ）
0DH	トークン抽出	６ＥＡ，ＢＨに解析したいテキストのアドレスを入れる。 出力。ＳＩ…次のトークンのアドレス 。ＢＬ…解析したトークン（80～ＦＥＨ） 。関数のときには， ＢＬ＝０ＣＨでＤＬにＦＦの次のコードが入る。 。ＵＳＲ関数のときは，ＤＸにＵＳＲの番号が入る。 。10～1FH の数字はＦＡＣにセットする。

<table>
<tr><th>DI</th><th>機　能</th><th>使用法・注意点</th></tr>
<tr><td>0DH
のつづき</td><td>トークン抽出</td><td>
<table>
<tr><th>BLの値</th><th>トークンの種別</th></tr>
<tr><td>04H</td><td>FN</td></tr>
<tr><td>0AH</td><td>USR</td></tr>
<tr><td>0CH</td><td>関数。FF　XX</td></tr>
<tr><td>0EH</td><td>REMの前の00(Nullコード)が
検出された。つまり，REM文だ。</td></tr>
<tr><td>10H</td><td>LIST．の．が検出された。
そのときの．の示す行のアドレス
はDXレジスタに入れられている。</td></tr>
</table>
</td></tr>
<tr><td>0EH</td><td>数式評価</td><td>6EA，6EBHに評価したい数式のアドレスを入れる。
例えば，テキスト，
A＊（10＋B）
↑
[6EA，6EB]
Bの値と10を足して，Aの値をかけた最後の結果を返してくる。
結果は，FACに入れられる。</td></tr>
<tr><td>0FH</td><td>テンポラリストリング
ディスクリプタの生成</td><td>一時的なストリングディスクリプタを作る。
例えば，ダイレクトモードで，
A$=“ABC”
とした場合など，画面をクリアすると，“ABC”は消えるので，文字列エリアに格納される。このときの一時的なストリングディスクリプタを作る。</td></tr>
<tr><td>10H</td><td>テキスト内部表現
→ソースイメージ変換</td><td>SI…ソースイメージにしたいテキスト（内部表現）の先頭アドレス
BX…ソースイメージを出力するバッファの先頭オフセットアドレス
（セグメントアドレスはDS＝60Hに固定されている。）</td></tr>
<tr><td>11H</td><td>行番号のバイナリ化</td><td>SI…ポインター（オフセットアドレス）
31　30　30　　　100
↑
出力　　SI
AX…バイナリ化された行番号</td></tr>
<tr><td>12H</td><td>ファイル番号の評価とチェック</td><td>[6EA，6EB] にファイル番号部分をさすアドレスを入れる
不適切なファイル番号のときは，
Bad file numberを出力し，ダイレクトモードに入る。</td></tr>
</table>

ＤＩ	機　能	使用法・注意点
12H のつづき	ファイル番号の評価とチェック	[1536,1537] …ファイルNo. [1538,1539] …File Control Blockアドレス
13H	ファイルディスクリプタのコード化	ドライブNo.の評価をし，ＣＯＭ，ＬＰＴ，ＳＣＲＮ，ＫＹＢＤのフラグをたてる。ファイル名をとり出す。 [6ＥＡ，6ＥＢ] に評価したいファイルディスクリプタ ↓ "2：ＤＥＭＯ．ＡＳＣ" 出力 152Ｃ～1534Ｈファイルネーム 152Ｂ　　ドライブNumber.
14H	デバイスタイプコードの取得	
15H	A00H番地の中身の番地をＣＡＬＬする。	A00H，A01H番地にオフセットアドレスを入れてよぶとインタープリンタ内のルーチンを直接よべる。
16H	変数アドレスの取得	[6ＥＡ，6ＥＢ] 変数名の先頭アドレスを入れる。 変数のアドレスは， 154Ｅ,154ＦＨ…オフセット 1550,1551Ｈ…セグメント に入って戻ってくる。
17H	ＦＡＣへのロード	[6ＥＡ，6ＥＢ] にアスキー数字列，変数名の先頭アドレスを入れる。 その値がＦＡＣに入る。
18H	ＦＡＣのストア	154Ｅ，154ＦＨ…オフセット，1550，1551Ｈ…セグメントで示されるアドレスに，ＦＡＣの値をストアする。
19H	ストリング値のロード	
1AH	コンマのスキップ	6EA，6EBHに解析中のアドレスを入れる。 次のトークンがコンマ（，2ＣＨ）でなければSyntax errorを表示して，コマンドまちになる。コンマならば，何もせず戻る。6EA，6EBHにはコンマの次のトークンをさしている。
1BH	指定行から実行を開始	実行したいテキストの１行の先頭のアドレスをＢＸレジスタに入れて呼ぶ。ＢＸでさすアドレスはまさしく行の先頭，つまり，リンクポインタをさしていなければならない。

DI	機　能	使用法・注意点
1CH	次の行を実行	6EC，6EDHに入れた先頭アドレスをもった行の次の行から実行する。 6EC，6EDHに入れる先頭アドレスは，1行の最初のリンクポインタをさしていること。
1DH	ＶＡＬ関数と同等	6EA，6EBHにアスキー文字列の先頭アドレスを入れる。結果は，ＦＡＣに入る。
1EH	ダイレクトモードエントリー	スタック等のイニシャライズをしてダイレクトモードに入る。ユーザー機械語の処理がおわったときに，ＢＡＳＩＣのコマンド待ちに戻るときのエントリーに使える。
1FH	プログラム編集時のステータスリセット	プログラムを編集すると，このルーチンがよばれる。 ＲＥＡＤポインタをテキストのTopにする。 添字の下限を0にする。 データスタックのクリア。ストリングエリアのクリア。 シンボルテーブルの初期化。エラーフラグのクリア。 Functionキーフラグのクリアなどをする。
20H	ＰＲＩＮＴ　ＵＳＩＮＧ用編集	
21H	値の編集（ＳＴＲ$）	
22H	スペースアイテムのスキップ	6EA，6EBHにセットしたアドレスから，スペースをスキップして，スペース以外のものが出てきたそのアドレスを6EA，6EBHにセットする。 スペースとは，アスキーコードの20H以外にも，内部表現のスペース1個～9個を示す01H～09Hも含む。
23H	文の終端判定	ＳＩにテキストのアドレスを入れて呼ぶ。 キャリーフラグが｛0…文の終端である 1…文の終端ではない
24H	バイナリ→10進数変換（ＳＴＲ$）	ＦＡＣにバイナリ表現の数値を入れてこのルーチンを呼ぶと，ＢＸレジスタによって指されるバッファ（セグメント60Hに固定）に，ソースイメージ（アスキーコードの数値表現）で格納する。

DI	機　能	使用法・注意点
25H	カレントデバイスへの出力	1840H…プリント先 { 3…プリンタ 4…CRT その他…ディスク，RS-232C等 1842，1843H…文字のバッファ CX…文字数
26H	DI＝1と同じ	Syntax errorの表示
27H	DI＝3と同じ	Type mismatchの表示
28H	DI＝2と同じ	Illegal function callの表示
29H	実数化	FAC＝CSNG（FAC）
2AH	倍精度化	FAC＝CDBL（FAC）
2BH	実数加算	FAC＝FAC1＋FAC
2CH	実数減算	FAC＝FAC1－FAC
2DH	実数乗算	FAC＝FAC1＊FAC
2EH	実数除算	FAC＝FAC1／FAC
2FH	実数比較	FAC1－FACをし，比較の結果は整数でFACに格納される。 { －1…FAC1＜FAC 0…FAC1＝FAC 1…FAC1＞FAC
30H	倍精度減算	FAC＝FAC1－FAC
31H	倍精度比較	実数比較と同じ
32H	バイナリ →8進数変換 （OCT$）	2バイトのバイナリ表現(FAC)をアスキー表現の8進数に変換バッファはBXでさす。
33H	バイナリ →16進数変換（HEX$）	DI＝32Hと同じだが，16進数として変換
34H	符号なし整数値 →10進変換	AXレジスタにバイナリ数(0～FFFFH)を入れて，よぶと，バッファにアスキー表現の数字(0～65535)を格納する。 バッファのポインタはBXレジスタである。 CXレジスタに文字数を出力する。

<table>
<tr><th>DI</th><th>機　能</th><th>使用法・注意点</th></tr>
<tr><td>35H</td><td>テキストアドレス
→行番号書き換え</td><td>テキスト中の飛び先等の実アドレス（内部表現の 0DH 2バイト数字）をすべて行番号に書き換える。
6D6Hに0を入れる。6D6Hは，テキスト内がすべて行番号に書き換えられたかどうかのフラグ。</td></tr>
<tr><td>36H</td><td>テキストから1項目抽出
（記号化して抽出）</td><td>6EA，6EBHにテキストアドレス。
出力はAL。
<table>
<tr><th>AL</th><th>意　味</th></tr>
<tr><td>0</td><td>00H(Null)REM文，文のおわり</td></tr>
<tr><td>1</td><td>0AH (LF) ラインフィード，"(ダブルクオテーション)内をスキップする。</td></tr>
<tr><td>2</td><td>01H～09Hで表わされた空白</td></tr>
<tr><td>3</td><td>0BH ×× ××　8進数</td></tr>
<tr><td>4</td><td>0CH ×× ××　16進数</td></tr>
<tr><td>5</td><td>0DH ×× ××　アドレス</td></tr>
<tr><td>6</td><td>0EH ×× ××　行番号</td></tr>
<tr><td>7</td><td>0FH ××　10～255の整数</td></tr>
<tr><td>8</td><td>漢字シフト</td></tr>
<tr><td>9</td><td>2バイト整数</td></tr>
<tr><td>A</td><td>4バイト単精度</td></tr>
<tr><td>B</td><td>8バイト倍精度</td></tr>
<tr><td>C</td><td>変数名</td></tr>
<tr><td>D</td><td>ステートメント</td></tr>
<tr><td>E</td><td>関数</td></tr>
</table>6EA，6EBHは次の項目をさす。</td></tr>
<tr><td>37H</td><td>指定行番号をもつテキストアドレスの取得</td><td>6EA，6EBHに行番号の先頭をさせる。
出力は，BXレジスタにアドレス。</td></tr>
<tr><td>38H</td><td>DI＝37Hと同じ</td><td>出力は，BXレジスタにアドレス。</td></tr>
<tr><td>39H</td><td>テキストのサーチ</td><td>入力　AXレジスタに行番号（バイナリ）
出力　キャリーフラグ…1そのような行番号はない…0有る。アドレスはBXレジスタに入っている。</td></tr>
<tr><td>3AH</td><td>数式存在のチェック</td><td>6EA，6EBHにテキストアドレス。
出力　キャリーフラグ…1数式はない
0数式はある</td></tr>
</table>

ＤＩ	機　能	使用法・注意点
3BH	ＣＲＴへの表示	セグメント60Ｈ，オフセット202Ｈ～のバッファの文字を，ＣＸレジスタで示される文字数だけ出力。 リモートＢＡＳＩＣプロトコルなど含む。
3CH	ＣＲＴへの表示（無条件）	上記と同じだが，必ずＣＲＴへ出力する。
3DH	１文字ＣＲＴへ表示	ＡＬレジスタの内容をＣＲＴへ出力する。
3EH	カーサのリセット	カーソルを行の左はしにもってくる。
3FH	ＣＲ，ＬＦの出力	リモートＢＡＳＩＣプロトコルを含む。
40H	ＣＲＴへＣＲ，ＬＦ出力	無条件にＣＲＴに C_R，L_F を出力する
41H	ＣＲＴカーサのリセット	ＣＲＴのカーソルを行のはしにもってくる。
42H	符号なし整数化	ＦＡＣ＝ＩＮＴ（ＦＡＣ）…０～65535
43H	添字評価	添字が範囲外ならエラーを出す。 （6EA，6EBHに添字の部分のテキストアドレス） 出力：ＢＰレジスタに配列の対応する部分のオフセットアドレスが入っている。
44H	ストリングエリアのゴミ集め	ガベージコレクションをする。
45H	符号なし整数の実数化	ＦＡＣ＝ＣＳＮＧ（ＦＡＣ） （ＤＩ＝42Ｈの逆）　└０～65535
46H	〝in ×××××〟表示（行番号）	×××××（行番号）の部分をＡＸレジスタに入れる。
47H	文の読みとばし	マルチステートメントの１文を読みとばす。 （6EA，6EBHにテキストアドレス）
48H	ＤＡＴＡ文の読みとばし	6EA，6EBHにＤＡＴＡ文のTopアドレスを入力すると，6EA，6EBHに次の行のTopが出力される。
49H	文字列定数の読みとばし	6EA，6EBHに↓ここをポイントさせる。 “文字列”： ↑──出力はここ
4AH	数式の評価 及び結果の整数化	6EA，6EBHに評価すべき数式のTopアドレスを入れる。 ＦＡＣに出力される。
4BH	キー，タイマ割り込みのセンス	キー入力やタイマ割り込みをセンスし，フラグをたてる。
4CH	ＣＯＭ，ＰＥＮ割り込みのセンス	ＲＳ-232Ｃやライトペン割り込みをセンスし，フラグをたてる。

DI	機　能	使用法・注意点
4DH	1行トランスレート	アスキーで書かれた1行のテキストをバイナリ表現にかえる。 ←長さバイト数CX→ アスキーの1行 ↑ 〔1406H〕 テキスト内に出力される。
4EH	メモリスイッチの内容取得	入力　BX(E2, E6, EA, EE, F2, F6, FA, FE) SW　1　2　3　4　5　6　7　8 出力　AL
4FH	未実装RAM アクセスチェック	AXレジスタにオフセット 750, 751Hにセグメント で指定したアドレスにRAMがあれば，何もせずに戻るが，RAMがなければ， Illegal function call を表示して，コマンド待ちになる。
50H	ストリングエリアの割り付け	CXレジスタに文字列の長さ（≦255）を入れて，呼ぶと，その分をストリングエリアに確保してくれる。 もし，空エリアがなければ， Out of string space のエラーが出てコマンド待ちになる。 ストリングエリアのポインタは6B4，6B5Hです。
51H	キーワードのサーチ	6EA，6EBHにテキストアドレスを入れて，ALに捜したいキーワードを入れておくと，6EA，6EBHにそのキーワードのあるアドレスを入れて帰る。 ないときは，ALレジスタに0が入っている。
52H	キーボードより1行入力	バッファは，202H～302Hです。 最後の2バイトにエンドマークとして0DH，0AHがつけ加えられる。

付録－3

ワークエリアー覧表

ワークエリアー覧表

(1)システム共通域　セグメント０：オフセット500Ｈ～5ＦＦＨ

アドレス	機　能　・　用　途
500	ＢＩＯＳ制御フラグ bit 3　ＶＳＹＮＣ通知 bit 4　拡張ＲＡＭ　有(1)，無(0) bit 5　キーボード オーバフロー bit 6　ＤＩＳＫＢＡＳＩＣのとき０ else 1 bit 7　スタートタイプが ウォームスタート(1)，コールド(0)
501	メモリサイズ 0…128KB（未拡張） 1…256KB 2…384KB 3…512KB 4…640KB
502～521	環状キーバッファ。 キーコードとして格納するので，２バイトで１文字です。
522，523	キーコード変換テーブルアドレス（オフセット） キーコードをＪＩＳ８単位コードに変換するときに使用する。
524，525	キーバッファ・ポインタ
526，527	キーバッファ・最終アドレス＋１
528	キーバッファ文字数
529	キーボード入力制御におけるエラー・リトライ回数
52Ａ～539	入力キーコードの押下状態に対応した96ビットの表。 「キー入力」の章の表を参照して下さい。
53Ａ	特殊キーの押下状態 bit 0…ＳＨＩＦＴ　bit 1…ＣＡＰ bit 2…カナ　bit 3…ＧＲＡＰＨ bit 4…ＣＴＲＬ
53Ｂ	ＣＲＴ行単位、ラスタ数－１

アドレス	機 能 ・ 用 途
53C	ＣＲＴ状態フラグ bit 0…ラインモード　0：25行，1：20行 bit 1…カラムモード　0：80カラム，1：40カラム bit 2…アトリビュートタイプ　1：バーティカルライン有効 bit 7…ＣＲＴタイプ　0：80ＣＲＴ，1：88ＣＲＴ
53D	ＢＩＯＳ用制御カウンタ
53E，53F	ＣＲＴ－ＢＩＯＳ制御に必要なコントロールブロックエリアのオフセットアドレス。
540，541	上記セグメントアドレス。
542，543	ＣＲＴＶ割り込みベクタ（No.10）の退避エリアのオフセットアドレス。
544，545	上記セグメントアドレス。
546	キャラクタジェネレータから読み出す文字フォントパターン bit 0…7×11（0），6×7（1） bit 1…ＡＮＫ（0），漢字（1）
547	ＧＤＣに設定するスクロールエリアの個数
548，549	表示画面ＶＲＡＭ上の開始アドレス（オフセット）
54A，54B	上記セグメントアドレス
54C	グラフィック画面のＣＲＴ状態 bit 6…0：80ＣＲＴ，1：88ＣＲＴ bit 7…0：表示中，　1：表示停止中
54D	ＧＤＣドット修正モード bit 1：bit 0　10進 0　0　0　ＲＥＰＬＡＣＥ 0　1　1　ＣＯＭＰＬＥＭＥＮＴ 1　0　2　ＣＬＥＡＲ 1　1　3　ＳＥＴ
54E，54F	ＧＤＣの線種パターン（ラインスタイル）
550～555	ＧＤＣのグラフィック文字パターン
556，557	ＲＳ－232Ｃ受信バッファ先頭アドレス（オフセット）
558，559	上記セグメントアドレス
55A	ＯＤＡ系プリンタのシフト状態をあらわす。　bit 0…0：ＳＯ，　1：ＳＩ bit 1…0：無シフト，1：シフト bit 2…0：割り込み無，1：有

アドレス	機能・用途
55B	ＲＳ－232Ｃの受信データシフト状態 ｛ＳＩ／ＳＯコード変換する（１），しない（０） 　シフト状態　ＳＩ（０），ＳＯ（１） bit 2…チャンネル２のシフト状態 bit 3…チャンネル１ bit 4…チャンネル０ bit 5…チャンネル２のＳＩ/ＳＯコード変換 bit 6…チャンネル１ bit 7…チャンネル０
55C	フロッピーディスク装置接続状況 （ビットがたっているとき接続されている） 物理ドライブ　論理ドライブ bit 0　0　1 bit 1　1　2　｝8インチ bit 2　2　3 bit 3　3　4 bit 4　0　1 bit 5　1　2　｝5インチ bit 6　2　3 bit 7　3　4
55D	5インチハードディスク装置接続状況 物理ドライブ　論理ドライブ bit 0　0　1 bit 1　1　2
55E	8インチフロッピーディスク割り込みフラグ 物理ドライブ　論理ドライブ bit 0　0　1 bit 1　1　2 bit 2　2　3 bit 3　3　4

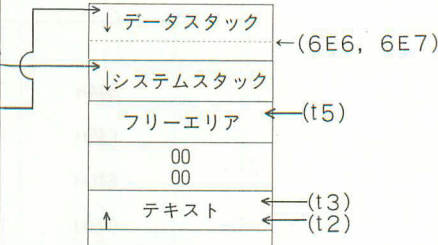
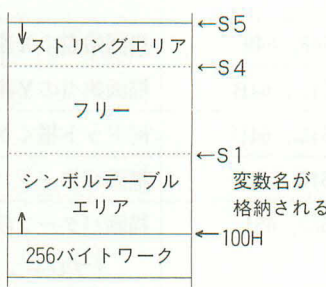
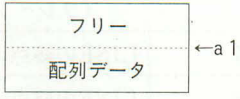
アドレス	機能・用途
55F	5インチハードディスク割り込みフラグ 物理ドライブ　論理ドライブ bit 0　0　1 bit 1　1　2
560	5インチフロッピーディスク装置のタイプ 00：未接続 ＥＦ：両面 ＦＦ：片面
561	5”フロッピーディスク両面装置のときのオペレーションモード bit 0…物理ドライブ0が { 0：片面処理モード / 1：両面処理モード } bit 1…物理ドライブ1 bit 2…物理ドライブ2 bit 3…物理ドライブ3
562, 563	タイムアウトチェック用カウンタ（5インチフロッピーディスク） 00：無条件ウェイト その他：×1 msecのウェイト
564～583	ディスクリザルト。フロッピーディスク割り込み情報 564Ｈ～56ＢＨ　ドライブ0 56CH～573Ｈ　ドライブ1 574Ｈ～57BH　ドライブ2 57CH～583Ｈ　ドライブ3

＋0	ST0
＋1	ST１又はPCN
＋2	ST２
＋3	C シリンダNo.(０～76)
＋4	H ヘッドNo.(０～１)
＋5	R セクタNo.(１～26)
＋6	N セクタ内データ長(０～３)
＋7	現在のシリンダNo.(０～76)

<table>
<tr><th>アドレス</th><th>機能・用途</th></tr>
<tr><td>584</td><td>システムディスクのアドレス
bit 0 ～bit 3　ＵＡ
bit 4 ～bit 7　ＤＡ</td></tr>
<tr><td>585</td><td>5インチハードディスクから送られるCompleteステータスバイト。
（ＢＩＯＳでのみ使用している）
bit 0………パリティーエラー
bit 1………エラー
bit 5 ～ 7…ＬＶＮ</td></tr>
<tr><td>586～589</td><td>5インチハードディスクコントローラから送られてくるエラー発生時のResultセンスバイトを格納する
<table><tr><td>＋0</td><td colspan="2">センスバイト</td></tr><tr><td>＋1</td><td>00d</td><td>論理アドレス(H)</td></tr><tr><td>＋2</td><td colspan="2">論理アドレス(M)</td></tr><tr><td>＋3</td><td colspan="2">論理アドレス(L)</td></tr></table></td></tr>
<tr><td>58A，58B</td><td>タイマＢＩＯＳで使用。
◦上位指定されたインターバルタイマ値
◦タイムアウト毎に減算するカウンター</td></tr>
<tr><td>58C，58D</td><td>未使用</td></tr>
<tr><td>58E～5BF</td><td>グラフィックＢＩＯＳ／ＬＩＯのＰＡＩＮＴ処理高速化制御用エリア。
次のＰＡＩＮＴ行のＧＤＣコマンドをスタックするエリア。
ＧＤＣ描画タイミングとＧＤＣコマンド作成のＣＰＵタイミングとを並行処理することによって高速化している。</td></tr>
<tr><td>5C0</td><td>ディップスイッチのコピーイメージ
bit 0 … 0 ：N－ＢＡＳＩＣ（86），　1 ：N_{88}－ＢＡＳＩＣ（86）
bit 1 … 0 ：ターミナルモード，　1 ：ＢＡＳＩＣモード
bit 2 … 0 ：80文字モード，　1 ：40文字モード
bit 3 … 0 ：25行モード，　1 ：20行モード
bit 4 … 0 ：メモリＳＷ有効，　1 ：無効</td></tr>
</table>

アドレス	機能・用途
5C1	受信したDELコードの扱いを指定する。 bit 0…0：そのままにする 1：メモリSW3のbit 7に従う メモリSW3のbit 7の意味は, 0：DELコード ┐ 1：NULコード ┘ に変換する
5C2, 5C3	GPIB BIOSとCALLERとの通信域のオフセットアドレス。
5C4, 5C5	上記セグメントアドレス。
5FE, 5FF	BASIC LIOのデータセグメントアドレス（通常60H）

(2)BASIC　LIO　ワークエリア　セグメント60H：オフセット0～1CFF

アドレス	機能・用途
0～201	0は，ＩＮＫＥＹ＄　1バイトバッファ ＰＲＩＮＴ文1行出力バッファ。2バイトで1文字。
202～24F	ＩＮＴ59～61の文字出力バッファ。1バイトで1文字。
314，315	テキストＶＲＡＭ1行目のＴＯＰアドレス
316	1行目のアトリビュート
317	80H以上…1行目と2行目がつながっている。 80H未満…1行目と2行目がつながっていない。
318，319 31A 31B	2行目のアトリビュート
31C，31D 31E 31F	3行目のアトリビュート
320，321 322 323	4行目のアトリビュート
324，325 326 327	5行目のアトリビュート
328，329 32A 32B	6行目のアトリビュート
32C，32D 32E 32F	7行目アトリビュート
330，331 332 333	8行目アトリビュート
334，335 336 337	9行目アトリビュート
338，359 33A 33B	10行目アトリビュート

アドレス	機能・用途
33C，33D 33E 33F	11行目アトリビュート
340，341 342 343	12行目アトリビュート
344，345 346 347	13行目アトリビュート
348，349 34A 34B	14行目アトリビュート
34C，34D 34E 34F	15行目アトリビュート
350，351 352 353	16行目アトリビュート
354，355 356 357	17行目アトリビュート
358，359 35A 35B	18行目アトリビュート
35C，35D 35E 35F	19行目アトリビュート
360，361 372 363	20行目アトリビュート
364，365 366 367	21行目アトリビュート

アドレス	機 能 ・ 用 途
368, 369 36A 36B	22行目アトリビュート
36C, 36D 36E 36F	23行目アトリビュート
370, 371 372 373	24行目アトリビュート
374, 375 376 377	25行目アトリビュート
378～42B	ファンクションキーバッファ 378H～389H F・1 load " 38AH～39BH F・2 auto 39CH～3ADH F・3 go to 3AEH～3BFH F・4 list 3C0H～3D1H F・5 run$^{C}_{R}$ 3D2H～3E3H F・6 save " 3E4H～3F5H F・7 key 3F6H～407H F・8 print 408H～419H F・9 edit . $^{C}_{R}$ 41AH～42BH F・10 cont $^{C}_{R}$ 各キーバッファ

＋0	状態フラグの窓 80H……ファンクションキーとして使う 20H……キー割り込みとして使う 00H……無視せよ。
＋1	格納文字数
＋2～10H	文字バッファ
＋11H	00(区切り)

アドレス	機能・用途
430, 431	画面の上限（通常0000Ｈ）consoleの第１引数
432, 433	画面の下限（通常0017Ｈ）
434, 435	画面をクリアするときの文字（通常0020Ｈ空白）
436, 437	カーソルｙ座標（25行時０～24，20行時０～19）
438, 439	カーソルｘ座標
43Ａ	画面をクリアするときのアトリビュートの値。 （通常Ｅ1Ｈ）
43Ｅ，43Ｆ	COLOR@　第２引数
440, 441	COLOR@　第１引数
442, 443	COLOR@　第４引数
444, 445	COLOR@　第３引数
449	bit１…ファンクションキー表示スイッチ bit７…カーソルスイッチ
44Ａ	割り込みフラグ
451	モニタモードフラグ
460, 461	画面の幅，１行の文字数
474	ＩＮＰＵＴ処理のときのフラグ
4A6	プリントルーチンの文字カウンター
4A8	ラインアトリビュート　ワークエリア
4E0～4FF	キー入力バッファ
503	ディスク台数
504	ファイル同時オープン数（０～ＦＨ） How many files　（０～15）？ に答えたときの値。デフォールトは，２
505	ＳＲＶ種別 １…N_{88}－ＢＡＳＩＣ（86） ２…５インチフロッピーディスク１ドライブ ｝Ｎ－ＢＡＳＩＣ（86） ３…５インチフロッピーディスク２ドライブ
506, 507	ディスクＰＩＯバッファ先頭オフセット
508, 509	ＤＣＢ群先頭オフセット
50Ａ，50Ｂ	ＦＣＢ群先頭アドレス
50Ｃ，50Ｄ	ＦＡＴバッファ先頭アドレス
50Ｅ，50Ｆ	ディスク定数テーブル先頭アドレス
514～51Ｄ	キーコードバッファ

<table>
<tr><th>アドレス</th><th colspan="2">機 能 ・ 用 途</th></tr>
<tr><td>570～5BF</td><td colspan="2">文字列転送バッファ</td></tr>
<tr><td>5C6</td><td>F・1の状態フラグ</td><td rowspan="10">80…ファンクションキー
20…キー割り込みON
00…ファンクションキーを無視する
ファンクションキーフラグの窓に書かれたフラグの値は，ここに格納される。</td></tr>
<tr><td>5C7</td><td>F・2の状態フラグ</td></tr>
<tr><td>5C8</td><td>F・3の状態フラグ</td></tr>
<tr><td>5C9</td><td>F・4の状態フラグ</td></tr>
<tr><td>5CA</td><td>F・5の状態フラグ</td></tr>
<tr><td>5CB</td><td>F・6の状態フラグ</td></tr>
<tr><td>5CC</td><td>F・7の状態フラグ</td></tr>
<tr><td>5CD</td><td>F・8の状態フラグ</td></tr>
<tr><td>5CE</td><td>F・9の状態フラグ</td></tr>
<tr><td>5CF</td><td>F・10の状態フラグ</td></tr>
<tr><td>620</td><td colspan="2">スクリーンモード</td></tr>
<tr><td>622</td><td colspan="2">ディスプレイページ</td></tr>
<tr><td>623</td><td colspan="2">フォアグラウンドカラー</td></tr>
<tr><td>624</td><td colspan="2">バックグラウンドカラー</td></tr>
<tr><td>626</td><td colspan="2">カラーパレット0のカラーコード</td></tr>
<tr><td>627</td><td colspan="2">カラーパレット1のカラーコード</td></tr>
<tr><td>628</td><td colspan="2">カラーパレット2のカラーコード</td></tr>
<tr><td>629</td><td colspan="2">カラーパレット3のカラーコード</td></tr>
<tr><td>62A</td><td colspan="2">カラーパレット4のカラーコード</td></tr>
<tr><td>62B</td><td colspan="2">カラーパレット5のカラーコード</td></tr>
<tr><td>62C</td><td colspan="2">カラーパレット6のカラーコード</td></tr>
<tr><td>62D</td><td colspan="2">カラーパレット7のカラーコード</td></tr>
<tr><td>62E～635</td><td colspan="2">VIEWの引数</td></tr>
<tr><td>アドレス</td><td colspan="2">640H～668HはグラフィックBIOS用コマンド指定バッファです。</td></tr>
<tr><td>640</td><td colspan="2">カラーパレットを設定する
下位3ビットに色の情報をかき込む
0…黒，1…青，2…赤，3…紫，4…緑，5…水色
6…黄，7…白</td></tr>
<tr><td>641</td><td colspan="2">ボーダーカラーを設定する
00…黒，10…青，20…赤，30…紫，40…緑
50…水色，60…黄，70…白</td></tr>
<tr><td>642</td><td colspan="2">1つのプレーンだけを処理するときのモード
0…おきかえ，1…XOR，2…NOT，3…OR</td></tr>
</table>

アドレス	機　能　・　用　途
643	描画方向の指定。
644～647	カラーパレット番号とカラーコードの対応
	上位４ビット / 下位４ビット
	644H: 6 / 7
	645H: 4 / 5
	646H: 2 / 3
	647H: 0 / 1
648, 649	描画始点のＸ座標
64A, 64B	描画始点のＹ座標
64C, 64D	何ドット描くかのドット数
64E, 64F	描画パターンバッファの先頭オフセットアドレス
650, 651	描画パターン読み出しバッファの先頭オフセット （プレーン１,４用）
652, 653	描画パターン読み出しバッファの先頭オフセット （プレーン２,５用）
654, 655	描画パターン読み出しバッファの先頭オフセット （プレーン３,６用）
656, 657	LINEの終点のＸ座標
658, 659	LINEの終点のＹ座標
65A	マスキングドット数
65C, 65D	円の半径
65E, 65F	フォントパターンの（たて方向のドット数）－１ ８×８のときは０
660, 661	ラインスタイル
662～667	８×８ドットグラフィック文字基本パターンバッファ。
668	描画タイプ １…直線，２…矩形，４…円孤
669～69F	グラフィックＢＩＯＳ用汎用ワークエリア
6A2, 6A3	配列データセグメントのセグメントベース（ａ）

アドレス	機能・用途
6A4，6A5	テキスト先頭アドレス（t_2）
6A6，6A7	テキスト最終アドレス（t_3）
6A8，6A9	未使用テキストエリアEND（t_5）
6AA，6AB	システムスタックの上限
6AC，6AD	データスタックの上限
	↓データスタック ←(6E6，6E7) ↓システムスタック フリーエリア ←(t5) 00 00 ↑ テキスト ←(t3) ←(t2)
6AE，6AF	シンボルテーブル先頭オフセット（100H）
6B0，6B1	シンボルテーブル最終オフセット（S_1）
6B2，6B3	ストリングエリアEND（S_5）
6B4，6B5	ストリングエリアポインタ（S_4）
	←S5 ↓ストリングエリア ←S4 フリー ←S1 シンボルテーブルエリア 変数名が格納される ↑ ←100H 256バイトワーク
6B6～6CF	それぞれ頭文字がA，B，C，D，…，Y，Zの変数の型 DEF INT（SNG，DBL，STR）で決定した値
6D0，6D1	配列ポインタ（a_1） フリー ←a 1 配列データ
6D2	TRONフラグ 0：TROFF，1：TRON
6D3	AUTOフラグ 0：OFF，1：ON
6D4	エラーフラグ（RESUMEでクリアされる）
6D5	
6D6	テキスト内の飛び先アドレスなどが，すべて行番号の状態になっている。 （そのとき0）
6D7	プロテクトフラグ
6D8，6D9	添字の下限。OPTION BASEで設定したもの。
6DA，6DB	AUTO行番号発生レジスタ
6DC，6DD	AUTOの増分

アドレス	機能・用途
6DE, 6DF	エラーが発生して止まったときの行番号。 ～エラー in ××××で使用する
6E0, 6E1	最後にエラーのおこった行番号。エラーが発生すると,6DE,Fの値をここにコピーする（ERL）
6E2	エラー番号（ERR）
6E3	
6E4, 6E5	実行中の行番号。ダイレクトモード時 FFFFH。
6E6, 6E7	データスタックポインタ。 GOSUB, FOR, WHILEのための戻り先, ループ回数カウンタ等の記憶スタック。
6E8, 6E9	実行中のテキスト内アドレス
6EA, 6EB	実行中のテキスト内アドレス。 インタプリタがコマンドを解析して各処理ルーチンに入いるとき，コマンドをさしている。 インタプリタ内ルーチンの利用（INT　C4H）の各処理ルーチンへのポインタとして使用されている。この場合,次のアイテムのアドレスがSIレジスタに入って戻ってくる。
6EC, 6ED	次の行の先頭アドレス
6EE, 6EF	6EC, 6EDのコピー
6F0, 6F1	データスタック処理ルーチンのワークエリア
6F2 6F3 6F4 6F5 6F6 6F7	
6F8, 6F9	実行中の行の先頭
6FA, 6FB	実行中のステートメントの先頭 （例）　PRINT “OK” ↑ ここをさしている
6FC, 6FD	データスタックのコピー
6FE	解析ルーチンのワーク

アドレス	機能・用途
702, 703	ＯＮ　ＳＴＯＰ　ＧＯＳＵＢフラグ
704, 705	ＯＮ　ＳＴＯＰ　ＧＯＳＵＢの飛び先アドレス
706, 707	ＯＮ　ＫＥＹ　ＧＯＳＵＢフラグ
708〜72D	ＯＮ　ＫＥＹ　ＧＯＳＵＢの飛び先アドレス
72E, 72F	ＯＮ　ＨＥＬＰ　ＧＯＳＵＢフラグ
730, 731	ＯＮ　ＨＥＬＰ　ＧＯＳＵＢの飛び先アドレス
732, 733	ＯＮ　ＴＩＭＥ　ＧＯＳＵＢフラグ
734, 735	ＯＮ　ＴＩＭＥ　ＧＯＳＵＢの飛び先アドレス
736, 737	ＯＮ　ＰＥＮ　ＧＯＳＵＢフラグ
738, 739	ＯＮ　ＰＥＮ　ＧＯＳＵＢの飛び先アドレス
74C〜74F	ＲＮＤの前の値
750, 751	ＤＥＦ　ＳＥＧ＝で指定したセグメント
756, 756	ＲＥＡＤのポインタ
758, 759	ＲＥＡＤのワーク
766	ＯＰＴＩＯＮ　ＢＡＳＥ使用フラグ
∫	
A00, A01	間接ＪＭＰ，ＣＡＬＬ用バッファ
A02, A03	インタプリタ内サブルーチンの利用（ＩＮＴ　Ｃ4Ｈ）のジャンプベクトルのあるオフセットアドレス
A04, A05	上記のセグメント·アドレス
A66〜	キーコード変換テーブル
A8A, A8B	タイマーのカウンター
∫	
B8F〜C8C	文字列 ＋０：文字数１バイト ＋１：文字列 ∫

アドレス	機能・用途
B8F～B91	C_R L_F キャリジリターン・ラインフィード
B92～B97	Break
B98～B9A	^C
B9B～B9D	^O
B9E～BAE	Terminal mode C_R L_F
BAF～BBD	Disk Version C_R L_F
BBE～BDE	NEC N－88 BASIC (86) version 1.0 C_R L_F
BDF～BEC	Bytes free C_R L_F
BED～BFF	? Redo from start C_R L_F
C00～C02	?
C03～C0B	ESCK!)!!ESCH
C0C～C11	Skip:
C12～C19	Found:
C1A～C20	Bad C_R L_F
C21～C30	Undefined line
C31～C38	KW= C_R L_F
C39～C48	0123456789ABCDEF
C49～C5F	How many files (0－15)?
C60～C65	Ok C_R L_F
C66～C8C	Random number seed (－32768 to 32767)?
1000～11FFHの512バイトはグラフィックLIO作業領域です。	
1000, 1001	DS SAVEエリア
1002, 1003	SS SAVEエリア
1004, 1005	SP SAVEエリア
1006, 1007	BX SAVEエリア
～	
1110～11FF	グラフィックパターン読み込みバッファ等に使用
～	

アドレス	機　能　・　用　途	
1380Ｈ～13ＦＦＨ（128バイト）はＧＣＯＰＹ用作業領域です。		
1400，1401	ディスクコードの先頭セグメント	h → 機械語プログラムセグメント m → 配列データセグメント （6A2、3）→ シンボルテーブルセグメント S → DISK CODE テキストセグメント t → システムエリア 00000
1402，1403	実装ＲＡＭ終端セグメント（ｈ）	
1404，1405	ユーザー機械語プログラムエリア 先頭のセグメント（ｍ）	
1406，1407	ＣＲＴ出力バッファの先頭オフセット （通常202Ｈ）	
1408，1409	中間言語バッファポインタ(通常1Ｂ00Ｈ)	
140A，140B	キーワード　インデックス　テー ブル先頭オフセットアドレス	
140C，140D	上記セグメントアドレス	
140E，140F	テキストエリアのセグメント(ｔ) 通常60Ｈ。	
1410，1411	シンボルテーブルセグメント(ｓ)	
1412，1413	テキストＶＲＡＭのセグメント（A000Ｈ）	
1414	ＦＡＣの型	
1415	bit 7…ＦＡＣの符号	
1416～141Ｄ	FAC (Floating Accumulator) 1416 1417 1418 1419 141A 141B 141C 141D 整数（141A～141B） 単精度（141A～141D） 倍精度（1416～141D）	
141Ｅ		
141Ｆ	ＦＡＣ1の符号…bit 7	

アドレス	機能・用途
1420～1427	FAC 1 1420 1421 1422 1423 1424 1425 1426 1427 倍精度（1420～1427） 整数（1424～1425） 単精度（1424～1427）
1429	8087スイッチ
14AE～14D5	ＵＳＲ関数の処理アドレスベクタテーブル +0 +1 オフセットアドレス +2 +3 セグメントアドレス 14AE ～14B1…ＵＳＲ0 14B2 ～14B5…ＵＳＲ1 14B6 ～14B9…ＵＳＲ2 14BＡ～14BD…ＵＳＲ3 14BＥ～14C1…ＵＳＲ4 14C2 ～14C5…ＵＳＲ5 14C6 ～14C9…ＵＳＲ6 14CＡ～14CD…ＵＳＲ7 14CＥ～14D1…ＵＳＲ8 14D2 ～14D5…ＵＳＲ9
14D6	ＡＵＴＯで同じ行番号があるかのフラグ。あれば，＊を表示する。
14E2	オーバーフロー，０除算エラーのフラグ
14ＥD	ＷＩＮＤＯＷ 使用フラグ
14ＥＥ～14ＦD	ＷＩＮＤＯＷ の引数。単精度で格納される。
14ＦＥ～1501	ＰＯＩＮＴ の座標
1502～1509	ＷＩＮＤＯＷ，ＶＩＥＷの横，縦の倍率（単精度）

アドレス	機能・用途
150A～	グラフィック　ルーチンワーク
152A	ファイルディスクリプタ DISK…0, CMT…1, COM…2 LPT…3, SCRN…4, KYBD…5
152B	ドライブ番号
152C～1534	ファイル名　152C～1531　6文字のファイル名 1532～1534　拡張子
1535	ＦＣＢのモード　　0…アスキー，2：プロテクト
1536	ファイル番号
1538	ファイルオープンフラグ
153A	ＷＩＤＴＨ　ＬＰＲＩＮＴ
154B	ＣＲＴに表示させないフラグ
154C，154D	オープンフラグ（ＴＥＲＭ）
154E，154F	変数のオフセットアドレス
1550,　1551	変数のセグメントアドレス
1582,　1553	ストリングスペースの限界
1584,　1585	使用済みストリングスペース
1586,　1587	FN関数の仮引数名テーブル定義ポインタ
1590	オープンフラグ（ＴＥＲＭ）
ｓ	
1593	拡張命令があるかのフラグ　0：無，1：有
1594	NEW　ON文実行フラグ
1598,　1599	ジャンプテーブル（オフセット）
159A，159B	ジャンプテーブル（セグメント）
159C，159D	拡張コマンド処理ルーチンのオフセット（ＣＭＤ等）
159E，159F	上記セグメント
15A0,　15A1	拡張コマンド（関数）処理ルーチンのオフセット （IEEE，STATUS）
15A2，15A3	上記　セグメント
15A8，15A9	ＣＯＮＳＯＬＥ　第1引数
15AA，15AB	ＣＯＮＳＯＬＥ　第1引数＋第2引数

アドレス	機 能 ・ 用 途
∫	
1800～ 1803	リストの速さ
∫	
1820	ＬＯＡＤ後ＲＵＮするかのフラグ
1824	0：アスキーセーブ，2：プロテクトセーブ
182A，B	ＢＳＡＶＥやＳＡＶＥでＳＡＶＥするエリアの先頭オフセット
182C，D	ＢＳＡＶＥやＳＡＶＥでＳＡＶＥするエリアの最終オフセット
1833	ＳＡＶＥするバイト数
∫	
1840	プリントアウト先 3：プリンタ　4：ＣＲＴ その他：ディスク，ＲＳ－232C（ＯＰＥＮで開いた先）
1842，1843	プリントバッファポインタ
∫	
1898～	ＰＲＩＮＴ　ＵＳＩＮＧ用編集バッファ
18A0～18B7	アスキーコードに直した数値の出力バッファ
18D0～18E7	アスキーコードに直した数値の出力バッファ
1A00，1A01	ＡＵＴＯ処理ルーチン用，ＡＵＴＯで発生する行番号
1A02，1A03	ＡＵＴＯ処理ルーチン用，ＡＵＴＯの増分
1A08，1A09	モニタのＳＰのＳＡＶＥエリア
1A14，1A15	モニタのＳＳのＳＡＶＥエリア
1A1C,1A1D	1行の文字数
1A1E,1A1F	ＤＳのＳＡＶＥエリア
1A20，1A21	ＳＳのＳＡＶＥエリア
1A22，1A23	ＳＰのＳＡＶＥエリア
1A24，1A25	h]C××××の値
1A5D,E	モニタのスタートセグメント値
∫	
1B00，1	ダイレクトモードで入力した中間コードのバッファ数
1B02，3	FFFFH（行番号）
1B04～1BFF	ダイレクトモードで入力したステートメントの中間コードバッファ
1C00～1CFF	ダイレクトモードで入力した文字列を一時格納するバッファ

(3)シンボルテーブルエリアのオフセット0～FFHのワークエリア

アドレス	機　能　・　用　途
0，1	ラベルテーブル先頭アドレス
2，3	単純変数テーブル先頭アドレス
4，5	配列変数ベクタテーブル先頭アドレス
3C～6F	頭文字A，B，C，…，Y，Zの変数のあるオフセットアドレス 3C，D…A　4A，B…H　56，7….N　64，5….U 3E，F…B　4C，D…I　58，9….O　66，7….V 40，1….C　4E，F…J　5A，B…P　68，8….W 42，3….D　50，1….K　5C，D…Q　6A，B…X 44，5….E　52，3….L　5E，F…R　6C，D…Y 46，7….F　54，5….M　60，1….S　6E，F…Z 48，9….G　62，3….T
70，71	単純変数エリアの大きさ
72，73	0000H
74～A7	それぞれ頭文字がA，B，C，…，Y，Zの配列名をもつ配列群の最終アドレス（下図） A　B　…　Z (72，73) =OOOOH　(74，75)　(76，77)　(A6，A7) 74，75…A　8E，8F…N 76，77…B　90，91…O 78，79…C　92，93…P 7A，7B…D　94，95…Q 7C，7D…E　96，97…R 7E，7F…F　98，99…S 80，81…G　9A，9B…T 82，83…H　9C，9D…U 84，85…I　9E，9F…V 86，87…J　A0，A1…W 88，89…K　A2，A3…X 8A，8B…L　A4，A5…Y 8C，8D…M　A6，A7…Z

付録－4

I／Oポート一覧表

ポートアドレス 16進	ポートアドレス 10進	内　　容
00H	0	マスター割り込みコントローラ（μPD8259A）（スレーブのI/Oポートは08H，0AH）

イニシャライズ・コマンド・ワード・フォーマット

ICW 1

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
OUT	0	0	0	1	LTIM	0	SNGL	IC4

IC4：通常1

（ICW1～ICW4の順でOUTする　ICW2～ICW4はI/Oポート02H）

LTIM｛1 レベル・トリガ入力／0 エッジ・トリガ入力

SNGL｛1 シングル／0 ノットシングル

オペレーション・コマンド・フォーマット

OCW 2

OUT	R	SL	EOI	0	0	L_2	L_1	L_0

R	SL	EOI	意　　味	
0	0	1	非特殊EOIコマンド	割込終了
0	1	1	特殊EOIコマンド＊	
1	0	1	非特殊EOIコマンド回転	自動回転
1	0	0	自動EOIモードで回転(SET)	
0	0	0	自動EOIモードで回転(CLEAR)	
1	1	1	特殊EOIコマンドで回転＊	特殊回転
1	1	0	優先セット・コマンド＊	
0	1	0	ノーオペレーション	

＊：L_0－L_2が用いられる

ポートアドレス		内　　容
16進	10進	

L_2	L_1	L_0		使用されるIRレベル	
0	0	0	0	IR0	IR8
0	0	1	1	IR1	IR9
0	1	0	2	IR2	IR10
0	1	1	3	IR3	IR11
1	0	0	4	IR4	IR12
1	0	1	5	IR5	IR13
1	1	0	6	IR6	IR14
1	1	1	7	IR7	IR15

(IR8～15はスレーブ)

OCW3

OUT

D_7							D_0
0	ESMM	SMM	0	1	P	RR	RIS

ESMM	SMM	
1	1	スペシャルマスクをセット
1	0	スペシャルマスクをリセット
0	1	何もしない
0	0	

P	
1	ポールコマンド
0	ノーポールコマンド

RR	RIS	
1	1	IRRリード
1	0	ISRリード
0	1	何もしない
0	0	

←ポート00Hをリード(IN)するときIRRを読むかISRを読むかを決める。

IN

IRR	IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0
ISR	IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0

IRRを読むかISRをよむかは、OUT命令でOCW3のD_1D_0で決める。

ポートアドレス		内容
16進	10進	
02 H	2	マスター割込コントローラ

ICW 2

OUT									
	0	0	0	0	1	0	0	0	マスター
	0	0	0	1	0	0	0	0	スレーブ(ポート0AH)

ICW 3

OUT	S7	S6	S5	S4	S3	S2	S1	S0	マスター
	1	0	0	0	0	0	0	0	

1：スレーブをもつ　　0：もたない

OUT	0	0	0	0	0	1	1	1	スレーブ(ポート0AH)

ICW 4

OUT									
	0	0	0	SFNM	BUF	1	0	1	マスター
	0	0	0	SFNM	BUF	0	0	1	スレーブ(ポート0AH)

SFNM { 1：スペシャルフリイオスティドモード
　　　 0：ノットスペシャルフリイオスティモード

BUF＝1

OCW 1

OUT									
	M_7	M_6	M_5	M_4	M_3	M_2	M_1	M_0	マスター
	M_{15}	M_{14}	M_{13}	M_{12}	M_{11}	M_{10}	M_9	M_8	スレーブ(ポート0AH)

IMRリード

IN									
	M_7	M_6	M_5	M_4	M_3	M_2	M_1	M_0	マスター
	M_{15}	M_{14}	M_{13}	M_{12}	M_{11}	M_{10}	M_9	M_8	スレーブ(ポート0AH)

ポートアドレス		内容									
16進	10進										

μPD8237A DMAコントローラ レジスタ選択表

双方向データバス IN, OUT

16進	10進	レジスタ	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
01H	1	CH-0 DMAアドレス	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
			A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	
03H	3	CH-0 カウンタ	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0	
			C_{15}	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8	
05H	5	CH-1 DMAアドレス	チャンネル0に同じ								
07H	7	CH-1 カウンタ	チャンネル0に同じ								
09H	9	CH-2 DMAアドレス	チャンネル0に同じ								
0BH	11	CH-2 カウンタ	チャンネル0に同じ								
0DH	13	CH-3 DMAアドレス	チャンネル0に同じ								
0FH	15	CH-3 カウンタ	チャンネル0に同じ								
11H	17	コマンドライト(プログラム時)	KS	DS	WS	PR	TM	CE	AH	MM	注)01000×00B をセット
		ステータス(読み出し時)	RO3	RO2	RO1	RO0	TC3	TC2	TC1	TC0	
13H	19	ライトリクエスト						RB	CS1	CS0	
15H	21	ライトシングルマスクレジスタビット						MK	CS1	CS0	
17H	23	ライトモード	MS1	MS0	ID	AT	TR1	TR0	CS1	CS0	
19H	25	クリアバイトポインタフリップフロップ									
		マスタクリア									
1BH	27	リードテンポラリレジスタ									
1DH	29	クリアマスタレジスタ									
1FH	31	ライトオールマスタレジスタビット					MB3	MB2	MB1	MB0	

10H～IEH(偶)はリザーブされている。

ポートアドレス		内　　　容
16進	10進	
1F H のつづき	31 のつづき	CH－0　5”固定ディスク CH－1　メモリリフレッシュ CH－2　8”フロッピィディスク CH－3　予備 MM　メモリメモリ　1許可0禁止　AH　CH－0アドレスホールド1許可0禁止 CE　コントローラ　1禁止0許可　TM　タイミング1圧縮0通常 PR　優先順位　1回転0固定　WS　ライト選択1拡張0遅れ DS　DREQアクティブ1 Low 0 High　KS　DACKアクティブ1 High 0 Low TC0～TC3　CH0～3がTCに到達 RQ0～RQ3　CH0～3がリクエスト

CS1	CS0	チャンネル選択
0	0	CH－0
0	1	CH－1
1	0	CH－2
1	1	CH－3

RB　リクエストビット　0リセット　1セット
MK　マスクビット　0クリア　1セット

TR1	TR0	転送モード
0	0	ベリファイ転送
0	1	ライト転送（I／O→メモリ）
1	0	リード転送（メモリ→I／O）
1	1	禁止

AT　オートイニシャライズ　0禁止　1許可
ID　アドレス　0インクリメント　1デクリメント

MS1	MS0	
0	0	デマンドモード
0	1	シングルモード
1	0	ブロックモード
1	1	カスケードモード

（注：01をセット）

ポートアドレス		内　　　容
16進	10進	
		MB 0～MB 3　CH 0～3マクビットを0クリア1セット
20H	32	カレンダ時計μPD1990

OUT

/	/	DI	CLK	STB	C_2	C_1	C_0

D I　入力データ　C L K　クロック　S T Bストローブ

C_2	C_1	C_0	
0	0	0	レジスタホールド
0	0	1	レジスタシフト
0	1	0	タイムセット及びカウンターホールド
0	1	1	タイムリード
1	0	0	禁 止
1	0	1	
1	1	0	
1	1	1	

注)リードデータはポート33Hのビット0

16進	10進	内容
		DMAバンク
23H	35	DMAチャンネル2用バンクライト
25H	37	DMAチャンネル3用バンクライト
27H	39	DMAチャンネル0用バンクライト
30H	48	R S232C（μP D8251A）データポート（I N/O U T）
32H	50	R S232Cコントロールポート（I N/O U T）

/	A19	A18	A17	A16

ポートアドレス		内　容
16進	10進	
32H のつづき	50 のつづき	非同期モード

ボーレート

B1	0	1	0	1
B2	0	0	1	1
	同期モード		X16	X64

キャラクタ長

L1	0	1	0	1
L2	0	0	1	1
	5ビット	6ビット	7ビット	8ビット

パリティイネーブル
1：イネーブル
0：ディスエイブル

パリティ指定
1：偶数
0：奇数

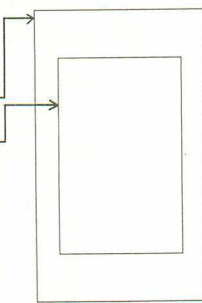
ストップビット数

S1	0	1	0	1
S2	0	0	1	1
	無　効	1ビット	$1\frac{1}{2}$ビット	2ビット

ポートアドレス		内　　　容
16進	10進	
32H のつづき	50 のつづき	同期モード

ポートアドレス		内　　容
16進	10進	
32H のつづき	50 のつづき	コマンドインストラクションの定義（OUT）

ポートアドレス		内　　容
16進	10進	
32H のつづき	50 のつづき	ステータスの読み出し（ＩＮ）

<table>
<tr><th colspan="2">ポートアドレス</th><th rowspan="2">内　　容</th></tr>
<tr><th>16進</th><th>10進</th></tr>
<tr><td>31H</td><td>49</td><td>システムポート　μPD8255Aのポート A
ディップ・スイッチ
<table><tr><td>IN</td><td>$\overline{SW8}$</td><td>$\overline{SW7}$</td><td>$\overline{SW6}$</td><td>$\overline{SW5}$</td><td>$\overline{SW4}$</td><td>$\overline{SW3}$</td><td>$\overline{SW2}$</td><td>$\overline{SW1}$</td><td>$\overline{SW0}$</td></tr></table></td></tr>
<tr><td>33H</td><td>51</td><td>システムポート　μPD8255Aのポート B
<table><tr><td>IN</td><td>$\overline{CI}$</td><td>$\overline{CS}$</td><td>$\overline{CD}$</td><td>INT3</td><td>CRT
TYPE</td><td>IM
CK</td><td>EM
CK</td><td>カレンダ時計
リードデータ</td></tr></table>
$\overline{CS}$　ＲＳ232Ｃ　ＣＳ信号
$\overline{CD}$　ＲＳ232Ｃ　ＣＤ信号
ＩＮＴ3　5″ＨＤ　ＩＮＴ信号
ＣＲＴＴＹＰＥ　1　640×400　0　640×200
ＩＭＣＫ　内部メモリパリティエラー
ＥＭＣＫ　外部メモリパリティエラー</td></tr>
<tr><td>35H</td><td>53</td><td>システムポートμPD8255Aのポート C
<table><tr><td>IN/OUT</td><td>／</td><td>$\overline{PSTBE}$</td><td>BAL</td><td>MC
KEN</td><td>BUZ</td><td>TX
RE</td><td>TX
EE</td><td>RX
RE</td></tr></table>
ＭＣＫＥＮ　メモリチェックイネーブル　エラー登録　1する0しない
（ポート33HのＩＭＣＫ，ＥＭＣＫに登録する）
ＢＵＺ　ブザー　0ＯＮ　1ＯＦＦ
ＴＸＲＥ　ＲＳ232ＣのＴＸＲＤＹによる割り込みのイネーブル
ＴＸＥＥ　ＲＳ232ＣのＴＸＥＭＰＴＹによる割り込みのイネーブル
ＲＸＲＥ　ＲＳ232ＣのＲＸＲＤＹによる割り込みのイネーブル</td></tr>
</table>

<table>
<tr><th colspan="2">ポートアドレス</th><th rowspan="2">内　　　容</th></tr>
<tr><th>16進</th><th>10進</th></tr>
<tr><td>37H</td><td>55</td><td>OUT　37H，92H…ポートA…IN
ポートB…IN
ポートC…OUT
（以上）にセットしている
<table>
<tr><th>D7 〜 D1</th><th>D0</th></tr>
<tr><td>0000000</td><td>RXRE F/FのON/OFF（1/0）</td></tr>
<tr><td>0000001</td><td>TXEE F/FのON/OFF（1/0）</td></tr>
<tr><td>0000010</td><td>TXR F/FのON/OFF（1/0）</td></tr>
<tr><td>0000011</td><td>BUZZER F/FのON/OFF（1/0）</td></tr>
<tr><td>0000100</td><td>MMチェックイネーブル（1/0）</td></tr>
</table></td></tr>
<tr><td>40H</td><td>64</td><td>プリンタインターフェースμPD8255A　ポートA（IN/OUT）
（セントロニクス）
IN/OUTデータポート</td></tr>
<tr><td>42H</td><td>66</td><td>セントロプリンタμPD8255A　ポートB（IN）
IN　bit 2…$\overline{\mathrm{BUSY}}$　（他のビットはdon't care）
bit 5…システムクロック　2：8MHz　0：5MHz</td></tr>
<tr><td>44H</td><td>68</td><td>セントロプリンタμPD8255AポートC（IN/OUT）
<table>
<tr><td>OUT</td><td>$\overline{\mathrm{PSTB}}$</td><td>0</td><td>0</td><td>0</td><td>IR8</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>IN</td><td>$\overline{\mathrm{PSTB}}$</td><td></td><td></td><td></td><td>IR8</td><td></td><td></td><td></td></tr>
</table>
$\overline{\mathrm{PSTB}}$…ストローブ，IR8…インターフェースから8259への
割り込み信号フラグ。</td></tr>
<tr><td>46H</td><td>70</td><td>セントロプリンタμPD8255Aのモード設定ポート（OUT）
OUT　、82H
OUT　0FH…$\overline{\mathrm{PSTB}}$のON
0EH…$\overline{\mathrm{PSTB}}$OFF
07H…IR8のON
06H…IR8のOFF</td></tr>
<tr><td>41H</td><td>65</td><td>キーボードインターフェースμPD8251A　（IN）
データポート</td></tr>
<tr><td>43H</td><td>67</td><td>キーボードインターフェースμPD8251A　（IN/OUT）</td></tr>
</table>

ポートアドレス		内　　容
16進	10進	
43H のつづき	67 のつづき	以下参照

モード/コマンド ライト

OUT

S_2	S_1	EP	PEN	L_2	L_1	B_2	B_1

モード5EHがセットされている

OUT

EH	IR	KBDE (RTS)	ER	RST (SBRK)	RxDE	RTY (DTR)	TxEN

コマンド

ステータス　リード

IN

／	／	FE	OE	PE	／	RDY	／

モード

S_2S_1	ストップビット長
00	無効
01	1ビット
10	1.5ビット
11	2ビット

L_2 L_1	キャラクタ長
00	5ビット
01	6ビット
10	7ビット
11	8ビット

ＥＰ　パリティ　1偶　0奇

ＰＥＮ　パリティイネーブル

1可0不

B_2 B_1	ボーレートファクター
00	同期モード
01	×1
10	×16
11	×64

コマンド

ＥＨ…無意味　ＩＲ…内部リセット　1でモードライトフォーマットに戻す。

ＫＢＤＥ（ＲＴＳ）…ＫＢ送信　1禁止　0許可

ＥＲ…エラーリセット　1ですべてのフラグ（ＰＥ，ＤＥ，ＦＥ）をクリア

ＲＳＴ（ＳＢＲＫ）…センドブレイクキャラクタ

R_xＥ…受信　1可　0不可

ＲＴＹ（ＤＴＲ）…リトライ　1無効0有効

T_xＥＮ…送信　1可　0不可

<table>
<tr><th colspan="2">ポートアドレス</th><th rowspan="2">内　　　容</th></tr>
<tr><th>16進</th><th>10進</th></tr>
<tr><td>43H
のつづき</td><td>67
のつづき</td><td>ステータス
ＦＥ…フレーミングエラー（終了で有効ストップビットが検出されない）
ＯＥ…オーバーランエラー（ＣＰＵが送信速度に追いつけない）
ＰＥ…パリティエラー（パリティエラー検出）
ＲＤＹ…インターフェース信号$\overline{\text{ＲＤＹ}}$と同じ（レディ）</td></tr>
<tr><td>50H</td><td>80</td><td>ＮＭＩ（Non Maskable Interrupt）フリップフロップ（ＯＵＴ）
ＲＡＭのパリティエラーが発生したときに割り込みを発生させない。
ＯＵＴ　dummyで設定される。</td></tr>
<tr><td>52H</td><td>82</td><td>ＮＭＩ（ＯＵＴ）
ＲＡＭのパリティエラーが発生したときに割り込みを
発生させる。ＯＵＴ　dummyで設定される。</td></tr>
<tr><td>51H</td><td>81</td><td>５インチフロッピィディスクインターフェース8255Ａ　ポートＡ（ＩＮ）
データ入力ポート</td></tr>
<tr><td>53H</td><td>83</td><td>５インチフロッピィディスクインターフェース8255Ａ ポートＢ（ＩＮ/ＯＵＴ）
データ出力ポート
ＩＮすると，出力したデータの確認ができる</td></tr>
<tr><td>55H</td><td>85</td><td>５インチフロッピィディスクインターフェース8255Ａポートｃ（ＩＮ/ＯＵＴ）
<table>
<tr><td>OUT</td><td>ATN</td><td>DAC</td><td>RFD</td><td>DAV</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>IN</td><td>ATN</td><td>DAC</td><td>RFD</td><td>DAV</td><td>／</td><td>DAC</td><td>RFD</td><td>DAV</td></tr>
</table></td></tr>
<tr><td>57H</td><td>87</td><td>５インチフロッピィディスクインターフェース8255Ａ　モードセット（ＯＵＴ）
<table>
<tr><td>モードセット</td><td>OUT</td><td>57H</td></tr>
<tr><td></td><td>ON</td><td>OFF</td></tr>
<tr><td>DAV</td><td>09H</td><td>08H</td></tr>
<tr><td>RFD</td><td>0BH</td><td>0AH</td></tr>
<tr><td>DAC</td><td>0DH</td><td>0CH</td></tr>
<tr><td>ATN</td><td>0FH</td><td>0EH</td></tr>
</table></td></tr>
</table>

<table>
<tr><th colspan="2">ポートアドレス</th><th rowspan="2">内　　　容</th></tr>
<tr><th>16進</th><th>10進</th></tr>
<tr><td>60H</td><td>96</td><td>CRT　コントローラμPD7220（テキスト）　（IN/OUT）
IN　ステータス
OUT　パラメータ</td></tr>
<tr><td>62H</td><td>98</td><td>CRTコントローラμPD7220（テキスト）　（IN/OUT）
IN　ライトペンデータ
OUT　コマンド</td></tr>
<tr><td>64H</td><td>100</td><td>コントローラμPD7220　（OUT）
CRTインタラプトリセット（ダミーをOUTする）</td></tr>
<tr><td>68H</td><td>104</td><td>コントローラμPD7220　（OUT）
モードフリップフロップコントロール
<table>
<tr><td>OUT</td><td>0</td><td>0</td><td>0</td><td>0</td><td>ADR2</td><td>ADR1</td><td>ADR0</td><td>DT</td></tr>
</table>
<table>
<tr><th rowspan="2">ADR2 ADR1 ADR0</th><th rowspan="2">機　能</th><th colspan="2">DT</th></tr>
<tr><th>1</th><th>0</th></tr>
<tr><td>0　0　0</td><td>アトリビュートセレクト</td><td>ATR4が
簡易グラフ</td><td>ATR4が
バーティカルライン</td></tr>
<tr><td>0　0　1</td><td>グラフィックモード</td><td>モノクロ</td><td>カラー</td></tr>
<tr><td>0　1　0</td><td>カラム幅</td><td>40字</td><td>80字</td></tr>
<tr><td>0　1　1</td><td>フォントセレクト</td><td>7×11</td><td>6×7</td></tr>
<tr><td>1　0　0</td><td>８８グラフモード</td><td>200本</td><td>その他</td></tr>
<tr><td>1　0　1</td><td>漢字アクセスモード</td><td>ドットマップ</td><td>コードアクセス</td></tr>
<tr><td>1　1　0</td><td>不揮発メモリモード</td><td>許可</td><td>禁止</td></tr>
<tr><td>1　1　1</td><td>表示許可</td><td>表示可</td><td>表示不可</td></tr>
</table></td></tr>
<tr><td>6CH</td><td>108</td><td>ボーダーカラー選択（OUT）
<table>
<tr><td>OUT</td><td>0</td><td>G</td><td>R</td><td>B</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>
<table>
<tr><th>GRB</th><th>色</th><th>RGB</th><th>色</th></tr>
<tr><td>0　0　0</td><td>黒</td><td>1　0　0</td><td>緑</td></tr>
<tr><td>0　0　1</td><td>青</td><td>1　0　1</td><td>シアン</td></tr>
<tr><td>0　1　0</td><td>赤</td><td>1　1　0</td><td>黄</td></tr>
<tr><td>0　1　1</td><td>マゼンタ</td><td>1　1　1</td><td>白</td></tr>
</table></td></tr>
</table>

61H～6FHはリザーブされている。

ポートアドレス 16進	ポートアドレス 10進	内　　容
		CRTマスタスライスコントローラμPD52611
70H	112	キャラクタ位置ライン数　PL
72H	114	ボディフェイスライン数　BL
74H	114	キャラクタ　ライン数　CL
76H	118	スムーススクロールライン数　SSL
78H	120	スクロールエリア上辺位置行数　SUR
7AH	122	スクロールエリア　行数　SDR

	CRT	25行	20行
PL	640×400	00H	1EH
	640×200	00H	1FH
BL	640×400	0FH	11H
	640×200	07H	08H
CL	640×400	10H	
	640×200	08H	

タイマーコントローラーμPD8253（IN/OUT）

16進	10進	内　　容
71H	113	カウンタ0のデータのリード/ライト（16ビットRL1 RL0で上位下位を指定する）
73H	115	カウンタ1のデータのリード/ライト
75H	117	カウンタ2のデータのリード/ライト
77H	119	モード指定

OUT	SC1	SC0	RL1	RL0	M2	M1	M0	BCD

SC1	SC0	
0	0	カウンタ0セレクト
0	1	カウンタ1セレクト
1	0	カウンタ2セレクト
1	1	使ってはいけない

RL1	RL0	
0	0	カウントラッチ
0	1	LSBのリード/ライト
1	0	MSBのリード/ライト
1	1	LSB、MSBの順にリード/ライト

M2	M1	M0	モード
0	0	0	0
0	0	1	1
×	1	0	2
×	1	1	3
1	0	0	4
1	0	1	5

BCD	
0	バイナリカウント(16ケタ)
1	BCDカウント(4ケタ)

ポートアドレス		内容
16進	10進	
		5インチ固定ディスクインターフェース
80H	128	データポート（IN/OUT）
82H	130	ステータス（IN）コントロール（OUT）

OUT	CHEN	NR DSW	SEL	0	RST	0	DM AE	INT E
1 IN	REQ	ACK	BSY	MSG	CXD	IXO		INT
0 IN	CT 1	CT 0	DT 02	DT 01	DT 00	DT 12	DT 11	DT 10

CHEN　Channel Enable　1でH/Aバスへのデータ出力が許可される。

NRDSW　Read Switch　ステータスビットが切り換えられる。

SEL　Select　1でH/AバスのSEL信号がオンになる。

RST　Reset　1→0でH/AバスのRST信号が400μSecのパルス幅でオンになる。

DMAE　Direct Memory Access Enable
1で　DMAモードとなる。

INTE　Interrupt Enable　1で割込みの発生が許可される。

ディップスイッチ

CT	DT0	タイプ名
0 0	0 0 0	PC98H31　1ユニット目5Mバイト
0 0	0 0 1	PC98H33　1ユニット目10Mバイト

DT1	タイプ名
0 0 0	PC98H32　2ユニット目増設用5Mバイト
0 0 1	PC98H34　2ユニット目増設用10Mバイト
∫	予　約
1 1 1	未　接　続

88H～8FHリザーブ

16進	10進	内容
90H	144	8インチフロッピーディスコントローラμPD765A ステータスポート

IN	RQM	DIO	NDM	CB	D3B	D2B	D1B	D0B

16進	10進	内容
92H	146	コマンド・パラメータ（OUT），リザルト（IN）

ポートアドレス		内　　容
16進	10進	
94H	148	ライト・コントロール（OUT）

I/O命令

命　令	I/Oポートアドレス	R/W	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	備　　考
ライト コマンド	92	W	←コマンド　レジスタ→	μPD756Aへコマンドをセットします。
ライト レジスタ	92	W	←　パ　ラ　メ－　タ　→	μPD756Aへのパラメータをセットします。NON－DMAモード時はFDへの書き込みデータもセットします。
リード　ステータス	90	R	←ステータスレジスタ→	μPD765Aからステータスを引き取ります。
リード　データ	92	R	←リザルトステータス→	μPD765Aからリザルトステータスを引き取ります。NON－DMAモード時にはFDから読み取ったデータも引き取ります。
ライト　ベース&カレント アドレス	09	W	←　ア　ド　レ　ス　→	DMA部のレジスタ
リード　カレント アドレス	09	R	←　ア　ド　レ　ス　→	DMA部のレジスタ
ライト　ベース&カレント カウント	0B	W	←　カ　ウ　ン　ト　→	DMA部のレジスタ
リード カレント　ワード カウント	0B	R	←　カ　ウ　ン　ト　→	DMA部のレジスタ
ライト　シングル マスク　レジスタビット	15	W	0　0　0　0　0　M　1　0	DMA部のレジスタ M：1　マスクオン M：0　マスクオフ
ライト　モード　レジスタ	17	W	0　1　0　0　m_1　m_2　1　0	DMA部のレジスタ m_1m_2：0 0　ベリファイ転送 m_1m_2：0 1　メモリライト転送 m_1m_2：1 0　メモリリード転送
クリア　バイト　ポインタ　フリップ フロップ	19	W	X　X　X　X　X　X　X　X	DMA部のレジスタ

ポートアドレス		内容
16進	10進	
94H のつづき	148 のつづき	(2) I/O命令

命令	I/O ポートアドレス	R/W	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	備考
ライト DMA チャンネル ＃2 バンク	23	W	X X X X ←バンク→	DMA部のレジスタ
ライト コントロール	94	W	R F S R X 1 X X X X T Y	μPD756Aの外部レジスタへのセット命令です。

X印：don't care

ライト　コントロール　レジスタ

D７ビット：ＲＳＴ…Reset

μPD765Ａ　LSIのRESET端子の入力信号となるレジスタでμPD765Aをイニシャライズするのに使用します。イニシャライズはμPD765Ａへのコマンド，パラメータの転送シーケンスやリザルトステータス転送シーケンスが乱れた時等に使用することができます。尚イニシャライズは電源投入直後及びRESETスイッチ押下時もハードウェアにて行なわれます。

D６ビット：ＦＲＹ…Forced Ready

D765ALSIのＲＤＹ端子の入力信号となるレジスタで，デバイスインタフェースのＲＤＹ信号と論理和されています。本ビットはデバイスの接続状態，デバイスの電源投入状態をチェックするために使用できます。フロッピーディスクとのインタフェースには，デバイスが接続されているか否か，電源が投入されているか否かの状態を直接示す信号線はありません。そこでデバイスにRecalibrate動作をさせてTrack ００信号が返って来たら，そのデバイスは接続かつ電源投入状態であると判定します。（デバイスは媒体が挿入されていなくてもRecalibrate動作を行う。）一方μPD765ＡはRDY端子がOFFであるとRecalibrateコマンドを実行しません。そこでＦＲＹビットによって強制的にＲＤＹ端子をONにすることによりRecalibrateコマンドを実行させることができます。

尚通常のSeekやRead/Writeコマンド実行時はＦＲＹビットはOFF

ポートアドレス		内　　　容
16進	10進	

にしておかねばなりません。さもないとNot Readyを検出できなくなります。

D４ビット：Must Be One

　セントロニクスインタフェース回路の$\overline{\text{PSTB}}$信号の許可フリップフロップです。Power On Resetやリセットスイッチ押下時のリセット動作後プリンタインタフェース（セントロニクス）のμPD8255のモードセット，および初期化が完了した後本フリップフロップを“１”にする必要があります。

　尚本フリップフロップはパワーオン/リセットスイッチのリセット時“０”がセットされます。

μPD765Ａ（FDC Floppy Disk Controller）

⑴レジスタ構成

　ＦＤＣ内にメインシステムとのインターフェース用レジスタとして，データレジスタ（ＤＡＴＡ）とステータスレジスタ（ＳＴＡＴＵＳ）を持っています。各レジスタはA_0，$\overline{\text{RD}}$，$\overline{\text{WR}}$制御信号によって選択され，それらの関係は下表の通りです。

I/Oポートアドレス	A_0	$\overline{\text{RD}}$	$\overline{\text{WR}}$	動　作
90	0	0	1	ステータス・レジスタ・リード
		×	0	禁　止
92	1	0	1	データ・レジスタ・リード
		1	0	データ・レジスタ・ライト

⑴データレジスタ（ＤＡＴＡ）

　ＦＤＣとメインシステム間で転送する各種情報(コマンド,パラメータ,データ,およびリザルトステータス)を一時的にストアする８ビットレジスタです。

⑵ステータスレジスタ（ＳＴＡＴＵＳ）

　ＦＤＣの状態を示す８ビットレジスタで，その構成を次表に示しま

ポートアドレス		内容
16進	10進	
94H のつづき	148 のつづき	す。メインシステムは任意の時点でその内容を読み取ることができます。

ステータスレジスタ

ビット番号	名称	略称	内容
D0	FD0 Busy	D0B	デバイス＃0がSEEKコマンドによるシーク動作を実行中であるが，シーク動作終了の割り込み要求を保留中であることを示します。
D1	FD1 Busy	D1B	デバイス＃1についてD0ビットの内容と同様
D2	FD2 Busy	D2B	デバイス＃2についてD0ビットの内容と同様
D3	FD3 Busy	D3B	デバイス＃3についてD0ビットの内容と同様
D4	FDC Busy	CB	FDCがCommand Phase, Result Phase,またはリード/ライト・コマンドのExecution Phaseを実行中であることを示します。このビットがセットされているときは，他のコマンドは受付けられません。
D5	Non-DMA MODE	NDM	FDCがNon－DMAモードでデータ転送中であり，メインシステムに対してサービスを要求していることを示します。
D6	DaTa Input/Output	DIO	データレジスタを介して転送するデータの方向を示します。0のときはメインシステムからFDCの方向，1のときはFDCからメインシステムの方向を示します。なお，データレジスタの状態はRQM（D7ビット）が示します。
D7	Request for Master	RQM	FDCからメインシステムへ転送すべきデータがデータレジスタにロードされていること，またはデータレジスタが空で，メインシステムからFDCへ転送するデータをデータレジスタに書き込んでもよいことを示します。データの方向を示すDIO（D6ビット）の状態により，次の働きをします。

ポートアドレス		内　　容
16進	10進	
94H のつづき	148 のつづき	（下表）

ビット番号	名　称	略称	内　　容
D 7	Request for Master	R Q M	D I O＝0のとき：　メインシステムからF D Cへデータを転送する場合で，メインシステムがF D Cのデータレジスタにデータをセットしたとき（$\overline{WR}$＝0）にR Q Mは0となり，F D Cがそのデータを引き取ったとき1となります。 D I O＝1のとき：　F D Cからメインシステムへデータ転送する場合で，F D Cがデータレジスタにデータをセットしたとき1となり，メインシステムがそのデータを引き取ったとき（$\overline{RD}$＝0）となります。

ポートアドレス		内　　容
16進	10進	
94H のつづき	148 のつづき	コマンドの出し方

コマンド		R/WA0 D7 ―――― D0	備　　考
READ DATA			SK：SKip DDAM
	C	W1 MT MF SK 0 0 1 1 0	MT：マルチトラック, MF：MFMモード
		× × × × × HD US1 US0	HD：ヘッド番号，US：デバイス番号
		←― C ―→	実行開始セクタのＩＤ情報
		←― H ―→	
		←― R ―→	
		←― N ―→	
		←― EOT ―→	トラック上の最終セクタ番号
		←― GPL ―→	Gap2の長さ(VFO SYNCを含まず)
		W1 ←― DTL ―→	処理すべきセクタ当りのデータ長
	E		データ転送
	R	R1 ←― ST0 ―→	実行終了時のステータス0
		←― ST1 ―→	実行終了時のステータス1
		←― ST2 ―→	実行終了時のステータス2
		←― C ―→	実行終了セクタのＩＤ情報
		←― H ―→	
		←― R ―→	
		R1 ←― N ―→	
READ DELETED DATA			SK：Skip DAM
	C	W1 MT MF SK 0 1 1 0 0	
		× × × × × HD US1 US0	READ DATAと同じ
		←― C ―→	
		←― H ―→	
		←― R ―→	
		←― N ―→	
		←― EOT ―→	
		←― GPL ―→	
		W1 ←― DTL ―→	
	E		データ転送
	R	R1 ←― ST0 ―→	READ DATAと同じ
		←― ST1 ―→	
		←― ST2 ―→	
		←― C ―→	
		←― H ―→	
		←― R ―→	
		R1 ←― N ―→	

ポートアドレス		内　　　容
16進	10進	
94H のつづき	148 のつづき	コマンドの出し方（下表）

コマンドの出し方

コマンド	R/WA0　D7 ———————— D0	備　　　考
READ ID		
C	W1　0　MF　0　0　1　0　1　0	
C	W1　×　×　×　×　×　HD　US1　US0	トラック上の最初に読んだエラーのないＩＤ情報をストアする。
E		
R	R 1 ←—— ST 0 ——→	READ　DATAと同じ
R	←—— ST 1 ——→	
R	←—— ST 2 ——→	
R	←—— C ——→	E－Phaseで読んだＩＤ情報
R	←—— H ——→	
R	←—— R ——→	
R	R 1 ←—— N ——→	
WRITE ID		
C	W1　0　MF　0　0　1　1　0　1	
C	×　×　×　×　×　HD　US1　US0	
C	←—— N ——→	データ長/セクタ
C	←—— SC ——→	セクタ数/トラック
C	←—— GPL ——→	Gapの長さ(VFO SYNCを含まず)
E	W 1 ←—— D ——→	データ領域に書き込むデータパターン　トラック上のセクタ数分のＩＤ情報をメインシステムより転送する。
R	R 1 ←—— ST 0 ——→	READ　DATAと同じ (ただし，C，H，Rは無意味)
R	←—— ST 1 ——→	
R	←—— ST 2 ——→	
R	←—— C ——→	
R	←—— H ——→	
R	←—— R ——→	
R	R 1 ←—— N ——→	

ポートアドレス		内容
16進	10進	
94H のつづき	148 のつづき	コマンドの出し方

コマンド		R/WA_0	D_7 ―――― D_0	備考
WRITE DATE	C	W1	MT MF 0 0 0 1 0 1	READ DATAと同じ
			× × × × × HD US1 US0	
			←― C ―→	
			←― H ―→	
			←― R ―→	
			←― N ―→	
			←― EOT ―→	
			←― GPL ―→	
		W1	←― DTL ―→	
	E			データ転送
	R	R1	←― ST0 ―→	READ DATAと同じ
			←― ST1 ―→	
			←― ST2 ―→	
			←― C ―→	
			←― H ―→	
			←― R ―→	
		R1	←― N ―→	
WRITE DELETED DATA	C	W1	MT MF 0 0 1 0 0 1	READ DATAと同じ
			× × × × × HD US1 US0	
			←― C ―→	
			←― H ―→	
			←― R ―→	
			←― N ―→	
			←― EOT ―→	
			←― GPL ―→	
		W1	←― DTL ―→	
	E			データ転送
	R	R1	←― ST0 ―→	READ DATAと同じ
			←― ST1 ―→	
			←― ST2 ―→	
			←― C ―→	
			←― H ―→	
			←― R ―→	
		R1	←― N ―→	

ポートアドレス		内　　容
16進	10進	
148 のつづき	94H のつづき	コマンドの出し方

コマンド		R/W	A_0	D_7 ――――― D_0	備　　考
READ DIAGNOSTIC		W	1	0　MF　0　0　0　0　1　0	
	C			×　×　×　×　×　HD　US1　US0	READ DATAと同じ
				←――― C ―――→	
				←――― H ―――→	
				←――― R ―――→	
				←――― N ―――→	
				←――― EOT ―――→	
				←――― GPL ―――→	
		W	1	←――― DTL ―――→	
	E				データ転送
	R	R	1	←――― ST0 ―――→	READ DATAと同じ
				←――― ST1 ―――→	
				←――― ST2 ―――→	
				←――― C ―――→	
				←――― H ―――→	
				←――― R ―――→	
		R	1	←――― N ―――→	
SCAN EQUAL	C	W	1	MT　MF　SK　1　0　0　0　1	READ DATAと同じ
				×　×　×　×　×　HD　US1　US0	
				←――― C ―――→	
				←――― H ―――→	
				←――― R ―――→	
				←――― N ―――→	
				←――― EOT ―――→	
				←――― GPL ―――→	
		W	1	←――― STP ―――→	比較すべきセクタのセクタ間隔1 or 2
	E				データ転送
	R	R	1	←――― ST0 ―――→	READ DATAと同じ
				←――― ST1 ―――→	
				←――― ST2 ―――→	
				←――― C ―――→	
				←――― H ―――→	
				←――― R ―――→	
		R	1	←――― N ―――→	

ポートアドレス 16進	ポートアドレス 10進	内　　容
94H のつづき	148 のつづき	コマンドの出し方

コマンド	R/W	A_0 D_7 ―― D_0	備　考
SCAN LOW OR EQUAL	W	1 MT MF SK 1 1 0 0 1	
C		× × × × × HD US1 US0	
		← C →	READ DATAと同じ
		← H →	
		← R →	
		← N →	
		← EOT →	
		← GPL →	
	W	1 ← STP →	比較すべきセクタのセクタ間隔1 or 2
E			データ転送
R	R	1 ← ST0 →	
		← ST1 →	
		← ST2 →	
		← C →	READ DATAと同じ
		← H →	
		← R →	
	R	1 ← N →	
SCAN HIGH OR EQUAL	W	1 MT MF SK 1 1 1 0 1	
C		× × × × × HD US1 US0	
		← C →	READ DATAと同じ
		← H →	
		← R →	
		← N →	
		← EOT →	
		← GPL →	
	W	1 ← STP →	比較すべきセクタのセクタ間隔1 or 2
			データ転送
R	R	1 ← ST0 →	
		← ST1 →	
		← ST2 →	
		← C →	READ DATAと同じ
		← H →	
		← R →	
	R	1 ← N →	

ポートアドレス		内　　　容
16進	10進	
94H のつづき	148 のつづき	コマンドの出し方

コマンドの出し方

コマンド	R/WA$_0$　D$_7$ ――――― D$_0$	備　　考
SEEK		
C	W1　0　0　0　0　1　1　1　1	
	W1　×　×　×　×　×　HD　US1　US0	
	←―― NCN ――→	新シリンダ番号
E		シーク動作
RECALI- BRATE		
BRATE C	W1　0　0　0　0　0　1　1　1	
	W1　×　×　×　×　×　0　US1　US0	
E		リカラブレイト動作
SENSE INT STATUS	W1　0　0　0　0　1　0　0　0	
R	R1　←―― ST0 ――→	
	R1　←―― PCN ――→	コマンド終了時のシリンダ番号
SENSE DEVICE C	W1　0　0　0　0　0　1　0　0	
STATUS	W1　×　×　×　×　×　HD　US1　US0	
R	R1　←―― ST3 ――→	デバイスの状態
SPECIFY		
C	W1　0　0　0　0　0　0　1　1	
	W1　← SRT →← HUT →	Step Rate Time,Head Unload Time
	W1　←― HLT ―→← ND	Head Load Time,Non−DMA Mode
Invalid		
C	W1　←― その他のコード ―→	
R	R1　←―― ST0 ――→	ST0＝80(10)

ポートアドレス		内　　容
16進	10進	
		カセットMTインターフェースμPD8251A
91H	145	データポート（IN/OUT）
93H	147	ステータス（IN）モード/コマンド（OUT）

IN	RxD	SYNDET	FE	OE	/	TxE	RxRDY	TxRDY

OUT	1	1	1	0	1	1	1	0	EEH
OUT	/	IR	/	ER	SBRK	RxE	/	TxEN	

16進	10進	内容
95H	149	コントロールレジスタ（OUT）

OUT	BS	CINH	CONT	/	/	TxEE	RxRE	TxRE

BSボーレート　　　0：600ボー　1：1200ボー

CINH　書き込みデータ禁止　1：ON　0：OFF

CONT　モータ　　　　　　1：ON　0：OFF

9XHの他のポートはリザーブ

16進	10進	内容
		CRTコントローラμPD7220（グラフィック）
A0H	160	ステータス（IN）　パラメータ（OUT）
A2H	162	データ（IN）　コマンド（OUT）
A4H	164	G－VRAMの切り換え（PC－9801F・E） 0……第1画面の表示 1……第2画面の表示
A6H	166	0……第1画面にアクセス 1……第2画面にアクセス
A8H	168	パレットレジスタNo
AAH	170	
ACH	172	
AEH	174	

上位	下位
3	7
1	5
2	6
0	4

文字パターンROMμPD23128

ポートアドレス		内　　容
16進	10進	
A1H	161	文字コード第2バイト（OUT）
A3H	163	文字コード第1バイト（OUT）
A5H	165	文字パターン行指定カウンタライト OUT ［／｜／｜L/R｜RC4｜RC3｜RC2｜RC1｜RC0］
A9H	169	文字パターンリード（IN） L/R＝1　　L/R＝0 RC3 RC2 RC1 RC0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0 0 1 0 1 0 1 1 0 0 1 1 1 1 0 0 0 1 0 0 1 1 0 1 0 1 0 1 1 1 1 0 0 1 1 0 1 1 1 1 0 1 1 1 1 RC4　未使用
B0H～BFH		リザーブ
C0H	192	ODAプリンタインターフェースμPD8255A ポートA（IN/OUT） データポート（INで確認できる）
C2H	194	リードシグナル1（IN）ポートB IN ［$\overline{RMR}$｜$\overline{ALM}$｜$\overline{MDL}$｜$\overline{DCN}$｜$\overline{IP1}$｜$\overline{IP2}$｜$\overline{IP3}$｜RDA］

ポートアドレス		内　　　容
16進	10進	
C4H	196	リードシグナル2（IN）ライトシグナル2（OUT） IN: $\overline{RDP}$ / RDA / $\overline{RMS}$ / ／ / IR6 / ／ / ／ / $\overline{IRT}$ OUT: 0 / 0 / 0 / 0 / 0 / 0 / 0 / $\overline{IRT}$
C6H	198	モードライト　ライトシグナル1（OUT）ポートC OUT　A2H （表：下記参照） RMR　プリンタがデータ受信可能状態 ALM　プリンタのハードエラー MDL　用紙残少，用紙切れ DCN　プリンタの電源ON IP3　タイムアウトになった。 RDA　データ送信可能状態
残りのCXHのポートはリザーブ		

	ON	OFF
$\overline{IRT}$	00H	01H
$\overline{RMS}$	0BH	0AH
INTE	0DH	0CH

ポートアドレス		内　　容
16進	10進	
D0H ∫ DFH	208 ∫ 223	リザーブ
E0H ∫	224 ∫	キーボード（スキャン方式） ただし　ＢＡＳＩＣのＩＮＰ命令のみ有効 機械語のＩＮ命令では無意味。 （ＢＡＳＩＣインタープリタ内でスキャン方式のシュミレートをしているのである。）

ポートアドレス		内　　　容
16進	10進	
EDH 〜 FFH	237 〜 255	リザーブ

付録－5

コマンド・ステートメント関数
処理アドレス一覧表

コマンド・ステートメント関数処理アドレス一覧表

AUTO	: 2274	LINE	: 3054	WHILE	: 7415
BSAVE	: 8C60	LOAD	: 8FC0	WEND	: 7444
BLOAD	: 8F68	LSET	: 7183	WRITE	: 9AE3
BEEP	: 515C	LFILES	: 6AB0	LIST	: 7661
CONSOLE	: 52AE	MOTOR	: 5189	SEG	: 3E12
COPY	: 6FD6	MERGE	: 8F2A	SET	: 4754
CLOSE	: 5245	MON	: 2A39	KINPUT	: 9AC3
CONT	: 22E0	NEXT	: 7387	SRQ	: 171E
CLEAR	: 2317	NAME	: 69A6	CMD	: 171E
CALL	: 9BC8	NEW	: 2A0B	IRESET	: 171E
COMMON	: 9C1C	NOT	: 3E12	ISET	: 171E
CHAIN	: 9C22	OPEN	: 51C7	POLL	: 171E
COM	: 24F1	OUT	: 7045	RBYTE	: 171E
CIRCLE	: 2D7C	ON	: 266E	WBYTE	: 171E
COLOR	: 2F6D	OPTION	: 5267		: 1719
CLS	: 3017	OFF	: 3E12		: 3E12
DELETE	: 2449	?	: 7FF5	>	: 3E12
DATA	: 22D9	PUT	: 3470	=	: 3E12
DIM	: 9D00	POKE	: 49EA	<	: 3E12
DEFSTR	: 9CC5	PSET	: 310D	+	: 3E12
DEFINT	: 9CBD	PRESET	: 3111	-	: 3E12
DEFSNG	: 9CC1	PAINT	: 3142	*	: 3E12
DEFDBL	: 9CB9	RETURN	: 2780	/	: 3E12
DSKO$	: 47A8	READ	: 9B79	^	: 3E12
DEF	: 7204	RUN	: 28A8	AND	: 3E12
ELSE	: 12C6	RESTORE	: 9BB0	OR	: 3E12
END	: 29FA		: 3E12	XOR	: 3E12
ERASE	: 9D1C	RESUME	: 293E	EQV	: 3E12
EDIT	: 2A51	RSET	: 71B5	IMP	: 1719
ERROR	: 2308	RENUM	: 2B46	MOD	: 1719
FOR	: 7270	RANDOMIZE	: 71E5		
FIELD	: 75E2	ROLL	: 3203		
FILES	: 6AB8	SCREEN	: 3227		
FN	: 3E12	STOP	: 16ED		
	: 3E12	SWAP	: 70E3		
GO TO	: 26E9	SAVE	: 8CBB		
GOSUB	: 26EF	SPC	: 3E12		
GET	: 3409	STEP	: 3E12		
HELP	: 251B	THEN	: 3E12		
INPUT	: 994E	TRON	: 75D2		
IF	: 2ABC	TROFF	: 75D9		
KEY	: 2569	TAB	: 3E12		
KILL	: 6A19	TO	: 3E12		
KANJI	: 3E12	TERM	: 6C1C		
LOCATE	: 53D6	USING	: 3E12		
L?	: 7FEE	USR	: 3E12		
LLIST	: 7668	WIDTH	: 5543		
LET	: 70D2	WAIT	: 7015		

※関数処理アドレスは第2章の56ページをご参照下さい。

注）これは1983年7月ごろの出荷されたPC-9801のものです。

付録－6

コルトロールコード一覧表

付録6　コントロールコード一覧表

(1) キーボード

16進	10進	対応するキー	N－BASIC (86)	N₈₈－BASI (86)
01	1	CTRL－A		ヘルプキーと同じ
02	2	CTRL－B	1つの前のワードへ戻る	(N－BASICと同じ)
03	3	CTRL－C	実行の中断（STOPの時）	実行の中断
04	4	CTRE－D		カーソル位置から1ワードを削除 (86)
05	5	CTRL－E	カーソル位置から後を消す	(N－BASICと同じ)
06	6	CTRL－F		1つ先のワードへ進む
07	7	CTRL－G	スピーカを鳴らす	(N－BASICと同じ)
08	8	CTRL－H	カーソル位置の左側の文字を削除する	(N－BASICとじ)
09	9	CTRL－I	水平タブ（8文字毎）	(N－BASICと同じ)
0A	10	CTRL－J	行を2つに分ける	ラインフィード、インサートモードで2行に分割
0B	11	CTRL－K	ホームポジション	(N－BASICと同じ)
0C	12	CTRL－L	テキスト画面クリア	(N－BASICと同じ)
0D	13	CTRL－M	キャリッジリターン	(N－BASICと同じ)
0E	14	CTRL－N	1つ先のワードへ進む	
0F	15	CTRL－O	ESCの後に押すことによりN₈₈－BASIC (86) と同じ働きを行なう	画面の表示を無効にする
12	18	CTRL－R	カーソル位置から右側を1文字分右へずらす。	インサートモードにする。
13	19	CTRL－S		実行を一時停止する
15	21	CTRL－U		1行キャンセル
18	24	CTRL－X		カーソルを行の最後に移す
1B	27	ESC	実行を一時停止する	
1C	28	→	カーソルを右へ移動	(N－BASICと同じ)
1D	29	←	カーソルを左へ移動	(N－BASICと同じ)
1E	30	↑	カーソルを上へ移動	(N－BASICと同じ)
1F	31	↓	カーソルを下へ移動	(N－BASICと同じ)

(2) 通　信

16進	10進	シンボル	シンボルの意味
0 0	0		null
0 1	1	S H	Start of Heading（ヘッディング開始）
0 2	2	S X	Start of Text（テキスト開始）
0 3	3	E X	End of Text（テキスト終了）
0 4	4	E T	End of Transmission（伝送終了）
0 5	5	E Q	Enquiry（問合わせ）
0 6	6	A K	Acknowledage（肯定応答）
0 7	7	B L	Bell（ベル、ブザー）
0 8	8	B S	Back Space（後退）
0 9	9	H T	Horizontal Tabulation（水平タブ）
0 A	1 0	L F	Line Feed（改行）
0 B	1 1	H M	Home (VT) Vertical Tabulation（垂直タブ）
0 C	1 2	C L	Clear (FF) Form Feed（改頁）
0 D	1 3	C R	Carriage Return（復帰）
0 E	1 4	S O	Sift-out（シフトアウト）
0 F	1 5	S I	Sift-in（シフトイン）
1 0	1 6	D E	Data Link Escape（伝送制御拡張）
1 1	1 7	D 1	Device Control1（装置制御１）
1 2	1 8	D 2	Device Control2（装置制御２）
1 3	1 9	D 3	Device Control3（装置制御３）
1 4	2 0	D 4	Device Control4（装置制御４）
1 5	2 1	N K	Negative Acknowledge（否定応答）
1 6	2 2	S N	Synchronous idle（同期信号）
1 7	2 3	E B	End of Transmission Block（伝送ブロック終了）
1 8	2 4	C N	Cancel（取消し）
1 9	2 5	E M	End of Medium（媒体終端）
1 A	2 6	S B	Substitute（文字置換）
1 B	2 7	E C	Escape（拡張）
1 C	2 8	→	(FS) File Separator（ファイル分離）
1 D	2 9	←	(GS) Group Separator（グループ分離）
1 E	3 0	↑	(RS) Record Separator（レコード分離）
1 F	3 1	↓	(US) Unit Separator（ユニット分離）

付録－7

エラーメッセージ一覧表

エラーメッセージ一覧表

1 ... NEXT without FOR
2 ... Syntax error
3 ... RETURN without GOSUB
4 ... Out of DATA
5 ... Illegal function call
6 ... Overflow
7 ... Out of memory
8 ... Undefined line number
9 ... Subscript out of range
10 ... Duplicate Definition
11 ... Division by Zero
12 ... Illegal direct
13 ... Type mismatch
14 ... Out of string space
15 ... String too long
16 ... String formula too complex
17 ... Can't Continue
18 ... Undefined user function
19 ... No RESUME
20 ... RESUME without error
21 ... Unprintable error
22 ... Missing operand
23 ... Line buffer overflow
24 ... ?
25 ... ?
26 ... FOR without NEXT
27 ... Tape read error
28 ... ?
29 ... WHILE without WEND
30 ... WEND without WHILE
31 ... Duplicate label
32 ... Undefined label
33 ... Feature not available
34 ... ?
35 ... ?
36 ... ?
37 ... ?
38 ... ?
39 ... ?
40 ... ?
41 ... ?
42 ... ?
43 ... ?
44 ... ?
45 ... ?
46 ... ?
47 ... ?
48 ... ?
49 ... ?
50 ... FIELD overflow
51 ... Internal error
52 ... Bad file number
53 ... File not found
54 ... File already open
55 ... Input past end
56 ... Bad file name
57 ... Direct statement in file

```
58 ... Sequential after PUT
59 ... Sequential I/O only
60 ... File not open
61 ... File write protected
62 ... Disk offline
63 ... ?
64 ... Disk I/O error
65 ... File already exists
66 ... ?
67 ... ?
68 ... Disk full
69 ... Bad allocation table
70 ... Bad drive number
71 ... Bad track/sector
72 ... Deleted record
73 ... Rename across disks
74 ... Illegal operation
```

付録－8

プリンタ機能一覧表

付録8　プリンタ機能一覧表(PC−8821/22, PC−8023)

分　類	ニーモニック	HEXコード	機　能	PC-8821/8822	PC-8023(C)
印字指令	CR	0D	バッファのデータを印字	○	○
改行	LF	0A	1行送り	○	○
垂直タブ	VT	0B	多行送り	○	○
フォームフィード	FF	0C	改ページ	○	○
拡大	SO	0E	拡大指令(8bit)	○	○
(8bit)	SI	0F	拡大解除(8bit)	○	○
セレクト	DC1	11	セレクト	○	○
ディセレクト	DC3	13	ディセレクト	○	○
拡大	DC2	12	拡大指令(7bit)	○	○
(7bit)	DC4	14	拡大解除(7bit)	○	○
水平タブ	HT	09	水平タブ移動	○	○
キャンセル	CAN	18	データのキャンセル	○	○
n行改行	US	1F	1〜15行の改行	○	○
VFU	−	−	タブ位置等の設定	○	○
印字方法	ESC, N	1B, 4E	HSパイカ	○	○
	ESC, P	1B, 50	プロポーショナル	○	○
	ESC, Q	1B, 51	コンデンス	○	○
	ESC, E	1B, 45	エリート	○	○
	ESC, H	1B, 48	HDパイカ	○	×
	ESC, K	1B, 4B	漢字	○	×
ドットスペース	ESC, SOH	1B, 01	1ドットスペース	○	○
	ESC, STX	1B, 02	2ドットスペース	○	○
	ESC, ETX	1B, 03	3ドットスペース	○	○
	ESC, EOT	1B, 04	4ドットスペース	○	○
	ESC, ENQ	1B, 05	5ドットスペース	○	○
	ESC, ACK	1B, 06	6ドットスペース	○	○
キャラクタモード	ESC, $	1B, 24	英数記号モード	○	○
	ESC, &	1B, 26	ひらがなモード	○	○
	ESC, #	1B, 23	内部グラフィックモード	○	×
ドット列印字モード	ESC, S	1B, 53	8bitドット列	○	○
	ESC, I	1B, 49	16bitドット列	○	×

分　類	ニーモニック	HEXコード	機　　能	PC-8821/8822	PC-8023(C)
ドット列印字モード	ESC, V	1B, 56	8 bitドット列リピート	○	×
	ESC, W	1B, 57	16bitドット列リピート	○	×
	ESC, F	1B, 46	ドットアドレッシング	○	×
キャラクタリピート	ESC, R	1B, 52	キャラクタリピート	○	×
強調文字	ESC, !	1B, 21	強調文字セレクト	○	○
	ESC,"	1B, 22	強調文字解除	○	○
印字モード	ESC,〕	1B, 5D	ロジカルシークモード	○	○
	ESC,〉	1B, 3E	片方向印字	○	×
	ESC,〔	1B, 5B	インクリメンタルモード	×	○
改行幅	ESC, A	1B, 41	1/6インチ改行モード	○	○
	ESC, B	1B, 42	1/8インチ改行モード	○	○
	ESC, T	1B, 54	N/120インチ改行モード	○	○
改行方向	ESC, f	1B, 66	順方向改行モード	○	○
	ESC, r	1B, 72	逆方向改行モード	○	○
水平タブ	ESC, (	1B, 28	水平タブセット	○	○
	ESC,)	1B, 29	水平タブ部分クリア	○	○
	ESC, 0	1B, 30	水平タブオールクリア	○	○
アンダーライン	ESC, X	1B, 58	アンダーライン開始	○	○
	ESC, Y	1B, 59	アンダーライン終了	○	○
レフトマージン	ESC, L	1B, 4C	印字開始位置への設定	○	○
リボン切換	ESC, C	1B, 43	リボン切替指定	○	×
外字のロード	ESC, *	1B, 2A	外字のロード	○	×
*ドット対応グラフィックドット数の切り換え	ESC, D	1B, 44	640ドットモード	○	×
	ESC, M	1B, 4D	960ドットモード	○	×

付録－9

キャラクターコード表

付録9　キャラクタコード表

ＡＳＣＩＩコード表（キャラクタセット）

上位４ビット→ / 下位４ビット↓

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		$^{D}{}_{E}$		0	@	P		p				ー	タ	ミ		
1	$^{S}{}_{H}$	$^{D}{}_{1}$	!	1	A	Q	a	q			。	ア	チ	ム		円
2	$^{S}{}_{X}$	$^{D}{}_{2}$	〃	2	B	R	b	r			「	イ	ツ	メ		年
3	$^{E}{}_{X}$	$^{D}{}_{3}$	#	3	C	S	c	s			」	ウ	テ	モ		月
4	$^{E}{}_{T}$	$^{D}{}_{4}$	$	4	D	T	d	t			、	エ	ト	ヤ		日
5	$^{E}{}_{Q}$	$^{N}{}_{K}$	%	5	E	U	e	u			・	オ	ナ	ユ		時
6	$^{A}{}_{K}$	$^{S}{}_{N}$	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		分
7	$^{B}{}_{L}$	$^{E}{}_{B}$	'	7	G	W	g	w			ァ	キ	ヌ	ラ		秒
8	$^{B}{}_{S}$	$^{C}{}_{N}$	(	8	H	X	h	x			ィ	ク	ネ	リ	♠	
9	$^{H}{}_{T}$	$^{E}{}_{M}$	)	9	I	Y	i	y			ゥ	ケ	ノ	ル	♥	
A	$^{L}{}_{F}$	$^{S}{}_{B}$	*	:	J	Z	j	z			ェ	コ	ハ	レ	♦	
B	$^{H}{}_{M}$	$^{E}{}_{C}$	+	;	K	[	k	{			ォ	サ	ヒ	ロ	♣	
C	$^{C}{}_{L}$	→	,	<	L	¥	l	¦			ャ	シ	フ	ワ	●	
D	$^{C}{}_{R}$	←	−	=	M	]	m	}			ュ	ス	ヘ	ン	○	
E	$^{S}{}_{O}$	↑	.	>	N	^	n	~			ョ	セ	ホ	゛		
F	$^{S}{}_{I}$	↓	/	?	O	_	o	DEL			ッ	ソ	マ	゜		DEL

付録−10

USING文フォーマット一覧表

付録10　ＵＳＩＮＧ文フォーマット一覧表

フォーマット	機　能	例
！	文字列の最初の文字だけ出力	PRINT USING "F. 1[！]";"abcde" F.1[a]
&　‥　& n文字	始めのn文字を左づめで出力	PRINT USING "F.2:[& &]";"abcde" F.2:[abcd]
@	１つの@に対して、１つの文字列を出力	PRINT USING "F.3:[Q]";"abcde" F.3:[abcde]
＃＃＃＃	数値を右づめで表示	PRINT USING "F.4:[＃＃＃＃]";123.456 F.4:[123]
＃＃＃＃．＃	小数点の位置を指定	PRINT USING "F.5:[＃＃＃＃．＃]";123.456 F.5:[123.5]
＋＃＃＃．＃	数値の前に符号（＋，－）をつける	PRINT USING "F.6:[＋＃＃＃．＃]";123.456 F.6:[＋123.5]
＃＃＃．＃＋	数値の後に符号（＋，－）をつける	PRINT USING "F.7:[＃＃＃．＃＋]";123.456, －123.456 F.7:[123.5＋]F.7:[123.5－] PRINT USING "F.8:[＃＃＃．＃－]";123.456, －123.456 F.8:[123.5]F.8:[123.5－]
＊＊＃＃＃＃．＃	空白部分を"＊"で埋める	PRINT USING "F.9:[＊＊＃＃＃＃．＃]"123.456 F.9:[＊＊＊123.5]
¥¥＃＃＃＃．＃	数値の直前に"¥"をつける	PRINT USING "F.10:[¥¥＃＃＃．＃]";123.456 F.10:[¥123.5]
＊＊¥＃＃．＃	＊＊と¥の両方の機能となる	PRINT USING "F.11:[＊＊¥＃＃．＃]";123.456 F.11:[＊¥123.5]
＃＃＃＃＃，	３桁毎に","で区切って出力	PRINT USING "F.12:[＃＃＃＃＃，]";1234.56 F.12:[1,235]
＃＃＃^^^^	数値を指数形式で出力	PRINT USING "F.13:[＃＃＃^^^^]";1234.56 F.13:[12E＋02]
＃＃＃＃_,＃＃＃＃	"_"に続く１文字を単に文字として出力	PRINT USING "F.14:[＃＃＃＃_，＃＃＃＃]" ;123.123 F.14:[123.123]

付録－11

Z80・8086ニーモニック対応表

ここでは、Ｚ－80の命令を8086の命令で対応させて一覧表を作成しました。Ｚ－80のプログラムを8086に移植したり、8086の学習に役立てて下さい。　なお、8086の命令への対応ではレジスタの保存は考慮していません。

Ｚ－80・8086レジスタ対応

Z−80			8086	
F	A	AF→AX	AH	AL
H	L	HL→BX	BH	BL
B	C	BC→CX	CH	CL
D	E	DE→DX	DH	DL

Z−80	8086
SP	SP
PC	IP
IX	SI
IY	DI

＊ＡＨとＦは、

　　ＬＡＨＦ…ＡＨ←Ｆ

　　ＳＡＨＦ…ＡＨ→Ｆ

という命令を通して対応します。

8ビット・ロード命令

```
 Z80                   8086
LD   r,r        ...   MOV  r,r
LD   r,n        ...   MOV  r,n
LD   r,(HL)     ...   MOV  r,[BX]
LD   r,(IX+d)   ...   MOV  r,d[SI]
LD   r,(IY+d)   ...   MOV  r,d[DI]
LD   (HL),r     ...   MOV  [BX],r
LD   (IX+d),r   ...   MOV  d[SI],r
LD   (IY+d),r   ...   MOV  d[DI],r
LD   (HL),n     ...   MOV  BYTE [BX],n
LD   (IX+d),n   ...   MOV  BYTE d[SI],n
LD   (IY+d),n   ...   MOV  BYTE d[DI],n
LD   A,(BC)     ...   PUSH BX
                      MOV  BX,CX
                      MOV  AL,[BX]
                      POP  BX
LD   A,(DE)     ...   PUSH BX
                      MOV  BX,DX
                      MOV  AL,[BX]
                      POP  BX
LD   A,(nn)     ...   MOV  AL,[nn]
LD   (BC),A     ...   PUSH BX
                      MOV  BX,CX
                      MOV  [BX],AL
                      POP  BX
LD   (DE),A     ...   PUSH BX
                      MOV  BX,DX
                      MOV  [BX],AL
                      POP  BX
LD   (nn),A     ...   MOV  [nn],AL
```

注）

レジスタ	r,r ′
Z−80	8086
B	CH
C	CL
D	DH
E	DL
H	BH
L	BL
A	AL

d……8ビットディスプレースメント

n……8ビットイミーディエイトデータ

n n……アドレス（16ビット）

16 ビット・ロード 命令

```
  Z-80                  8086
LD   dd,nn       ･･･  MOV  dd,nn
LD   IX,nn       ･･･  MOV  SI,nn
LD   IY,nn       ･･･  MOV  DI,nn
LD   HL,(nn)     ･･･  MOV  BX,[nn]
LD   dd,(nn)     ･･･  MOV  dd,[nn]
LD   IX,(nn)     ･･･  MOV  SI,[nn]
LD   IY,(nn)     ･･･  MOV  DI,[nn]
LD   (nn),HL     ･･･  MOV  [nn],BX
LD   (nn),dd     ･･･  MOV  [nn],dd
LD   (nn),IX     ･･･  MOV  [nn],SI
LD   (nn),IY     ･･･  MOV  [nn],DI
LD   SP,HL       ･･･  MOV  SP,BX
LD   SP,IX       ･･･  MOV  SP,SI
LD   SP,IY       ･･･  MOV  SP,DI
PUSH qq          ･･･  PUSH qq
PUSH AF          ･･･  LAHF
                      PUSH AX
PUSH IX          ･･･  PUSH SI
PUSH IY          ･･･  PUSH DI
POP  qq          ･･･  POP  qq
POP  AF          ･･･  POP  AX
                      SAHF
POP  IX          ･･･  POP  SI
POP  IY          ･･･  POP  DI
```

注）

レジスタdd	
Z−80	8086
BC	CX
DE	DX
HL	BX
SP	SP

レジスタqq	
Z−80	8086
BC	CX
DE	DX
HL	BX
AF	AX

n n……16ビットイミーディエイトデータ、アドレス

エクスチェンジ 命令

```
 Z80                       8086
EX    DE,HL      ・・・    XCHG DX,BX
EX    AF,AF′     ・・・    XCHG AX,[nn]   ┐
EXX              ・・・    XCHG CX,[nn1]  │ 注)
                           XCHG DX,[nn2]  │
                           XCHG BX,[nn3]  ┘
EX    (SP),HL    ・・・    XCHG BX,AX
                           CLI
                           XCHG BP,SP
                           XCHG [BP],AX
                           XCHG BP,SP
                           STI
                           XCHG BX,AX
EX    (SP),IX    ・・・    XCHG SI,AX
                           CLI
                           XCHG BP,SP
                           XCHG [BP],AX
                           XCHG BP,SP
                           STI
                           XCHG SI,AX
EX    (SP),IY    ・・・    XCHG DI,AX
                           CLI
                           XCHG BP,SP
                           XCHG [BP],AX
                           XCHG BP,SP
                           STI
                           XCHG DI,AX
```

注)8086では裏レジがないのでメモリとのエクスチェンジになります。

ブロック転送命令

```
 Z80                       8086
LDI              ・・・    CLD                 ;[ES:DI]←[DS:SI]
                           MOVSB               ;DI=DI+1
                                               ;SI=SI+1
                                               ;CX=CX-1
LDIR             ・・・    CLD                 ;[ES:DI]←[DS:SI]
                           REP   MOVSB         ;DI=DI+1
                                               ;SI=SI+1
                                               ;CX=CX-1 until CX=0
LDD              ・・・    STD                 ;[ES:DI]←[DS:SI]
                           MOVSB               ;DI=DI-1
                                               ;SI=SI-1
                                               ;CX=CX-1
LDDR             ・・・    STD                 ;[ES:DI]←[DS:SI]
                           REP   MOVSB         ;DI=DI-1
                                               ;SI=SI-1
                                               ;CX=CX-1 until CX=0
```

ブロックサーチ命令

```
 Z80                 8086
CPI         ...   CLD                  ;[DS:SI]-[ES:DI]
                  CMPSB
CPIR        ...   CLD                  ;[DS:SI]-[ES:DI]
                  REP   CMPSB           SI=SI+1:DI=DI+1
                                        until CX=0
CPD         ...   STD                  ;[DS:SI]-[ES:DI]
                  CMPSB
CPDR        ...   STD                  ;[DS:SI]-[ES:DI]
                  REP   CMPSB           SI=SI-1:DI=DI-1
                                        until CX=0
```

8ビット算術論理演算命令

```
 Z80                 8086
ADD  r          ...   ADD  AL,r
ADD  A,n        ...   ADD  AL,n
ADD  A,(HL)     ...   ADD  AL,[BX]
ADD  A,(IX+d)   ...   ADD  AL,d[SI]
ADD  A,(IY+d)   ...   ADD  AL,d[DI]
ADC  A,r        ...   ADC  AL,r
ADC  A,n        ...   ADC  AL.n
ADC  A,(HL)     ...   ADC  AL,[BX]
ADC  A,(IX+d)   ...   ADC  AL,d[SI]
ADC  A,(IY+d)   ...   ADC  AL,d[DI]
SUB  r          ...   SUB  AL,r
SUB  n          ...   SUB  AL.n
SUB  (HL)       ...   SUB  AL,[BX]
SUB  (IX+d)     ...   SUB  AL,d[SI]
SUB  (IY+d)     ...   SUB  AL,d[DI]
SBC  A,r        ...   SBC  AL,r
SBC  A,n        ...   SBC  AL,n
SBC  A,(HL)     ...   SBC  AL,[BX]
SBC  A,(IX+d)   ...   SBC  AL,d[SI]
SBC  A,(IY+d)   ...   SBC  AL,d[DI]
AND  r          ...   AND  AL,r
AND  n          ...   AND  AL,n
AND  (HL)       ...   AND  AL,[BX]
AND  (IX+d)     ...   AND  AL,d[SI]
AND  (IY+d)     ...   AND  AL,d[DI]
OR   r          ...   OR   AL,r
OR   n          ...   OR   AL,n
OR   (HL)       ...   OR   AL,[BX]
OR   (IX+d)     ...   OR   AL,d[SI]
OR   (IY+d)     ...   OR   AL,d[DI]
XOR  r          ...   XOR  AL,r
XOR  n          ...   XOR  AL,n
XOR  (HL)       ...   XOR  AL[BX]
XOR  (IX+d)     ...   XOR  AL,d[SI]
XOR  (IY+d)     ...   XOR  AL,d[DI]
CP   r          ...   CMP  AL,r
CP   n          ...   CMP  AL,n
CP   (HL)       ...   CMP  AL,[BX]
CP   (IX+d)     ...   CMP  AL,d[SI]
CP   (IY+d)     ...   CMP  AL,d[DI]
INC  r          ...   INC  r
```

```
INC  (HL)      ···  INC  [BX]
INC  (IX+d)    ···  INC  BYTE d[SI]
INC  (IY+d)    ···  INC  BYTE d[DI]
DEC  r         ···  DEC  r
DEC  (HL)      ···  DEC  BYTE [BX]
DEC  (IX+d)    ···  DEC  BYTE d[SI]
DEC  (IY+d)    ···  DEC  BYTE d[DI]
```

１６ビット算術論理演算命令

```
 Z80                 8086
ADD  HL,ss     ···  ADD  BX,ss
ADC  HL,ss     ···  ADC  BX,ss
SBC  HL,ss     ···  SBB  BX,ss
ADD  IX,pp     ···  ADD  SI,pp
ADD  IY,rr     ···  ADD  DI,rr
INC  ss        ···  INC  ss
INC  IX        ···  INC  SI
INC  IY        ···  INC  DI
DEC  ss        ···  DEC  ss
DEC  IX        ···  DEC  SI
DEC  IY        ···  DEC  DI
```

注)

レジスタss	
Z—80	8086
BC	CX
DE	DX
HL	BX
SP	SP

レジスタpp	
Z—80	8086
BC	CX
DE	DX
IX	SI
SP	SP

レジスタrr	
Z—80	8086
BC	CX
DE	DX
IY	DI
SP	SP

アキュムレータ操作命令

```
 Z80                 8086
DAA            ···  DAA
NEG            ···  NEG  AL
CPL            ···  NOT  AL
CCF            ···  CMC
SCF            ···  STC
```

ＣＰＵコントロール命令

```
 Z80                 8086
NOP            ···  NOP
HALT           ···  HLT
DI             ···  CLI
EI             ···  STI
```

ローテート・シフト命令

```
 Z80                       8086
RLC  A           ...   ROL   AL,1
RL   A           ...   RCL   AL,1
RRC  A           ...   ROR   AL,1
RR   A           ...   RCR   AL,1
RLC  r           ...   ROL   r,1
RLC  (HL)        ...   ROL   BYTE [BX],1
RLC  (IX+d)      ...   ROL   BYTE d[SI],1
RLC  (IY+d)      ...   ROL   BYTE d[DI],1
RL   r           ...   RCL   r,1
RL   (HL)        ...   RCL   BYTE [BX],1
RL   (IX+d)      ...   RCL   BYTE d[SI],1
RL   (IY+d)      ...   RCL   BYTE d[DI],1
RRC  r           ...   ROR   r,1
RRC  (HL)        ...   ROR   BYTE [BX],1
RRC  (IX+d)      ...   ROR   BYTE d[SI],1
RRC  (IY+d)      ...   ROR   BYTE d[DI],1
RR   r           ...   RCR   r,1
RR   (HL)        ...   RCR   BYTE [BX],1
RR   (IX+d)      ...   RCR   BYTE d[SI],1
RR   (IY+d)      ...   RCR   BYTE d[DI],1
SLA  r           ...   SAL   r,1
SLA  (HL)        ...   SAL   BYTE [BX],1
SAL  (IX+d)      ...   SAL   BYTE d[SI],1
SAL  (IY+d)      ...   SAL   BYTE d[IY],1
SRA  r           ...   SAR   r,1
SRA  (HL)        ...   SAR   BYTE [BX],1
SRA  (IX+d)      ...   SAR   BYTE d[SI],1
SRA  (IY+d)      ...   SAR   BYTE d[DI],1
SRL  r           ...   SHR   r,1
SRL  (HL)        ...   SHR   BYTE [BX],1
SRL  (IX+d)      ...   SHR   BYTE d[SI],1
SRL  (IY+d)      ...   SHR   BYTE d[DI],1
RLD              ...   MOV   AL,[BX]
                       XCHG  CX,DX
                       MOV   CL,4
                       ROL   AL,CL
                       MOV   CL,4
                       ROL   AX,CL
                       XCHG  CX,DX
                       MOV   [BX],AH
RRD              ...   MOV   AH,[BX]
                       XCHG  CX,DX
                       MOV   CL,4
                       ROR   AX,CL
                       MOV   CL,4
                       ROR   AL,CL
                       XCHG  CX,DX
                       MOV   [BX],AX
```

ビット操作命令

```
  Z80                      8086
BIT  b,r          ・・・  TEST  r,b'           ;BIT 1,A
BIT  b,(HL)       ・・・  TEST BYTE [BX],b'     →TEST AL,02H
BIT  b,(IX+d)     ・・・  TEST BYTE d[SI],b'
BIT  b,(IY+d)     ・・・  TEST BYTE d[DI],b'
SET  b,r          ・・・  OR    r,b'           ;SET 1,(IX+1)
SET  b,(HL)       ・・・  OR    BYTE [BX],b'    →OR BYTE 01H[DI],02H
SET  b,(IX+d)     ・・・  OR    BYTE d[SI],b'
SET  b,(IY+d)     ・・・  OR    BYTE d[DI],b'
RES  b,r          ・・・  AND   r,b'           ;RES 1,(HL)
RES  b,(HL)       ・・・  AND   BYTE [BX],b'    →AND BYTE [BX],0FDH
RES  b,(IX+d)     ・・・  AND   BYTE d[SI],b'
RES  b,(IY+d)     ・・・  AND   BYTE d[DI],b'
```

注）b' は上の例を参照して算出して下さい

ジャンプ・コール・リターン命令

```
  Z80                      8086
JP   nn           ・・・  JMP   nn
JR   nn           ・・・  JMPS  nn
JP   NZ,nn        ・・・  JE    NEXT
                          JMP   nn
                   NEXT:---
JR   Z,nn         ・・・  JNE   NEXT
                          JMP   nn
                   NEXT:---
JP   NC,nn        ・・・  JB    NEXT
                          JMP   nn
                   NEXT:---
JP   C,nn         ・・・  JNB   NEXT
                          JMP   nn
                   NEXT:---
JP   PO,nn        ・・・  JP    NEXT
                          JMP   nn
                   NEXT:---
JP   PE,nn        ・・・  JNP   NEXT
                          JMP   nn
                   NEXT:---
JP   P,nn         ・・・  JS    NEXT
                          JMP   nn
                   NEXT:---
JP   M,nn         ・・・  JNS   NEXT
                          JMP   nn
                   NEXT:---
JR   NC,nn        ・・・  JNB   nn
JR   Z,nn         ・・・  JE    nn
JR   NZ,nn        ・・・  JNE   nn
JP   (HL)         ・・・  JMP   BX
JP   (IX)         ・・・  JMP   SI
JP   (IY)         ・・・  JMP   DI
DJNZ nn           ・・・  LOOP nn  (カウンタは CXレジ)
CALL nn           ・・・  CALL nn
CALL NZ,nn        ・・・  JE NEXT
                          CALL nn
```

```
CALL NC,nn      ･･･  JB   NEXT
                     CALL nn
                NEXT:---
CALL C,nn       ･･･  JNB  NEXT
                     CALL nn
                NEXT:---
CALL PO,nn      ･･･  JP   NEXT
                     CALL nn
                NEXT:---
CALL PE,nn      ･･･  JNP  NEXT
                     CALL nn
                NEXT:---
CALL P,nn       ･･･  JS   NEXT
                     CALL nn
                NEXT:---
CALL M,nn       ･･･  JNS  NEXT
                     CALL nn
                NEXT:---
RET             ･･･ RET
RET  NZ         ･･･  JE   NEXT
                     RET
                NEXT:---
RET  Z          ･･･  JNE  NEXT
                     RET
                NEXT:---
RET  NC         ･･･  JB   NEXT
                     RET
                NEXT:---
RET  C          ･･･  JNB  NEXT
                     RET
                NEXT:---
RET  PO         ･･･  JP   NEXT
                     RET
                NEXT:---
RET  PE         ･･･  JNP  NEXT
                     RET
                NEXT:---
RET  P          ･･･  JS   NEXT
                     RET
                NEXT:---
RET  M          ･･･  JNS  NEXT
                     RET
                NEXT:---
RETI            ･･･  IRET
```

入出力命令

```
 Z80                  8086
IN   A,n        ･･･  IN   AL,n
IN   r,(C)      ･･･  MOV  DH,0
                     MOV  AH,AL
                     IN   AL,DX
                     MOV  r,AL
                     MOV  AL,AH
INI             ･･･  MOV  DH,0
                     MOV  AH,AL
                     IN   AL,DX
                     MOV  [BX],AL
                     INC  BX
```

```
                 MOV   AL,AH
                 DEC   CL
INIR         ... MOV   DH,0
                 MOV   AH,AL
           LOOP: IN    AL,DX
                 MOV   [BX],AL
                 INC   BX
                 DEC   CL
                 JNE   LOOP
                 MOV   AL,AH
IND          ... MOV   DH,0
                 MOV   AH,AL
                 IN    AL,DX
                 MOV   [BX],AL
                 MOV   AL,AH
                 DEC   BX
                 DEC   CL
INDR         ... MOV   DH,0
                 MOV  AH,AL
           LOOP: IN    AL,DX
                 MOV   [BX],AL
                 DEC   BX
                 DEC   CL
                 JNE   LOOP
                 MOV   AL,AH
OUT  n,A         OUT   n,AL
OUT  (C),r   ... MOV   DH,0
                 MOV   AH,AL
                 MOV   AL,r
                 OUT   DX,AL
                 MOV   AH,AL
OUTI         ... MOV   DH,0
                 MOV   AH,AL
                 MOV   AL,[BX]
                 OUT   DX,AL
                 MOV   AL,AH
                 INC   BX
                 DEC   CL
OTIR         ... MOV   DH,0
                 MOV   AH,AL
           LOOP: MOV   AL,[BX]
                 OUT   DX,AL
                 INC   BX
                 DEC   CL
                 JNZ   LOOP
                 MOV   AL,AH
OUTD         ... MOV   DH,0
                 MOV   AH,AL
                 MOV   AL,[BX]
                 OUT   DX,AL
                 MOV   AL,AH
                 DEC   BX
                 DEC   CL
OTDR         ... MOV   DH,0
                 MOV   AH,AL
           LOOP: MOV   AL,[BX]
                 OUT   DX,AL
                 DEC   BX
                 DEC   CL
                 JNZ   LOOP
                 MOV   AL,AH
```

索 引

B

BASICマシン語ルーチン ……265
BASICプログラム復活……46
BASICプログラム復活の原理……48
BIOSコール……95
BIOSコマンド一覧表 ……179
BP(オフセットアドレス) ……183

C

CALL……121・248・262
CALL文の引数……255
CIRCLE……304
CLOSE……176
CLS……75
CMD……319
CMTインターフェイス……154
CMTとデータ転送の仕様……154
COLOR……81
COPY……208
CPUアドレス……59
CRT……222・275

D

DATA文作成プログラム……326
DEF SEG……251
DEF USR……251
DIPスイッチ……58
DMA(Direct Memory Access) ……178
DSKF関数……173
DSKI$……275
DSKO$……275

E

ES(セグメントアドレス) ……183
Edit……242

F

FAT(File Allocation Table) ……171

G

G CIRCLE ……112
G CLS ……111
G COLOR……110
G COLOR 2……111
G COPY ……119
GDC(7200)……59
G END ……269
GET#1……176
G GET ……116
G INIT ……106
G LINE……112
G PAINT 1 ……114
G PAINT 2 ……115
G PUT 1 ……117
G PUT 2 ……117
G ROLL……118
G SCREEN……107
G STEP ……111
G VIEW……109
G-VRAM……233・269

I

ID情報……181
IDセクタ……171
INI KEY……327
INKEY$でカーソル表示 ……307・331
INP関数……146
IPL……184
I/Oポート……102

K

KI,KO……226
KINPUT……227
KPLOAD……301

L

LINE……89
LINEのBIOSワークエリア……90
LIO(Logical Input Output) ……104
LOGO……300
LPRINT……211・219
LPT1:……274・275

N

N_{88}BASIC……22
N_{88}DISK BASIC……171・248
N_{88}DISK BASIC(86)……184
N_{88}日本語BASIC(86)……298

O

ODA……16・217
OPEN……171・176
OUT PUT……307・333

P

PC9801E……305
PC9801F……297
PEEK……264

PEEK POKEでの引数……264
POINT……300
POKE……264
PRINT……219
PRINT＃1……176
PUT＃1……176

R
RAMのメモリマップ……17
REM文の効率……307
ROLL……75
ROLL200＆ROLL400……269
RS232C……289

S
SCREEN……101

T
TABキーとTAB関数……70
TXTCOP……211

U
UFO……132
USR……121
USR関数……261
USR関数・CALL文とマシン語……248
USR関数の引数……252

V
VARPTR……276
VIEWポート……77
V RAM……59・194

W
WIDTH……58

ア
アキュームレター(FAC)……252
アスキーSAVE……25
新しいコマンドを作る……236
アトリビュート……276
アドレス空間……14
アドレスサーチ……270
アドレスの表し方……15

イ
イニシャライズ……190
イニシャライズ(初期化)……217
インターリーブ・13……174
インタプリタのキー入力……138
インタラプトコール……186
インデックスポインタ……26
インベーダー……62
インベーダーパターン……100

エ
エラーメッセージディスプレイ……314
円孤を描く(CIRCLE)……93

オ
オフセット……194

カ
カーテンコール・クリア……103
拡張グラフィック画面……297
拡張ステートメント……297・300
カセット……275
カセットから読み込み……157
カセットへの書き込み……157
カセットファイル……154
片面・両面アクセス……194
画面を縦に2分割……64
カラーグラフィックコピー……212
カラーコード……77
カラーコードの指定……78
カラーパレット……77
カラーパレットのI／Oポート……78
カラーパレットの初期化……80
カラーパレットの情報……79
簡易グラフ……61
簡易グラフィック……62
漢字……226
漢字ROMと日本語BASIC……297
漢字ROMボード……226
漢字・JISコード対応表……236
漢字フォント……235
関数処理アドレス……50
ガベージコレクション……46
画面コピー機能……208
画面の退避・復活……67

キ
キューアドレス……136
キューバッファ……136
キー・スキャン方式……146
キーコード……137
キースキャン対応表……148
キーセンス……148

キーセンス比較表……152
キー入力……136
キー入力バッファ……136
キー入力方法……151
キーバッファクリア……334
キーワード……25
キー割り込みON……142
行番号0……307・308

ク

クラスタ……172
クロック8MHz……298
グラフィックBIOSとGDC……81
グラフィックVRAM……14
グラフィック画面……74・208
グラフィックスのワークエリア……82
グラフィックパターンを描く……98

ケ

結果の戻し方……261

コ

5インチディスク……189
5インチ片面倍密度……167・168
5インチ固定ディスク(5MB)……168
5インチ固定ディスク(10MB)……169
5インチ両面倍密度……167
5インチ両面倍密度倍トラック……167
コール……190
高速書き込みモード……101
高速画面クリア……77
高速グラフィックスローダー……283
高速リスト……307・332
コピー機能一覧……208
コミュニケーションプログラム……295
小文字・大文字変換……266

サ

サーフェス……172
最大値を求める……267
最大値サーチ……267
3Dパッケージ……121
サウンドビープ……265

シ

シャトルクリア……103
シーク……186
シーケンシャルファイル……176
システムフォーマット……174
シリンダ……179
シリンダとヘッド……179
シンボルテーブルセグメント……33
時間表示プログラム……335

ス

スクロールアップ……77
ステータス……217
ストップキーチェックルーチン……149
ストリングディスクリプタ……44・253

セ

整数型・単精度型……40
セクタ……172
セクタ長……180
セグメント……15・194
セグメント・ポインタ……19
セントロニクス……16
セントロニクス系プリンタ……217
専用ケーブルの作り方……289
1200ボー……293

ソ

増設RAM……15
属性(アトリビュート)……60

タ

タイマー(時間待ち)……251
タイマールーチン……160
単純変数テーブル……35・37・38
単密度……183
単密モード……183

チ

中間言語……24
中間言語コード……24
中間言語テーブル……27
中間コード表……33
直線と箱型を描く……89

ツ

通信速度……292
通信モードの指定……289・291

テ

テキストVRAM……14
テキストVRAMのアドレス……59
テキスト画面……58・208
テキスト画面の2ページ目……67

テキスト画面のコピー……211
テキストサーチ……340
テイルポインタ……139
ディスクアドレスとクラスタとの交換……169
ディスクの物理構造……166
ディスクマップ……166
ディスクBIOSコマンド……178・189
ディスクファイル……166
ディレクトリ……169
ディプスィッチ……15
データ書き込み……156
データスタック……19
データの高速セーブ・ロード……161
データファイル……155
データフォーマット……154
デバイス種別……181
デバイス名の一覧……274

ト
トラック……172
ドットを読み出す……86
ドット情報セット……84
ドットの指定数……85

ナ
内部ルーチン……155

ニ
2点と箱型の適切な方向……92
入出力ファイル……273
入力ファイル……273

ハ
配列変数テーブル……39
バッファカウンタ……139
バイト数……183
倍密度……183
バブルクリア……103
8086リセット……330
8インチIDリーダー……203
8インチ両面倍密度……166
バリアブルリスト……359
バーティカル・ファイルズ……359

ヒ
標準ディスク……174
引数(パラメータ)……248
ひらがなの表示……68

フ
ファイルコントロールブロック(FCB)……276
ファイルネームソート……199
ファイルの属性……169
ファイルバッファ……273
ファイルバッファアドレス……278
ファイルバッファ使用例……280
ファーコール(セグメント間コール)……251
ファンクションキー……139・227
ファンクションキーの構造……139
ファンクションキーの初期化……143
ファンクションキーの退避・復活……145
フォーマット……187
フォントパターン……229
不揮発性メモリ……15
フラグ……25
物理アドレス……15
物理フォーマット……174
プリンタ出力……208
プレーン……74
プログラムの格納状態……22
プログラムの転送……289・293
プログラムファイル……154

ヘ
ヘッド……179
変数テーブル……35
変数でファイル指定……273
変数のリンクポインタ……37

ホ
ボーダーカラー……80
ボーレート……293

マ
マシン語によるセーブ・ロード……156
マシン語ファイル……155

ミ
未使用コマンドを使用……318

メ
メモリスイッチ……15
メモリマップ……14・275

モ
文字型配列変数……43
文字コード……137
文字列を逆に表示……268

文字列エリア……………………………………………44

ユ

ユーザーマシン語……………………………………20
ユーティリティ………………………………………340
ユニット番号…………………………………………181

ラ

ラベルテーブル………………………………………33
ラベルの登録…………………………………………33
ランダムテクニック…………………………………307
ランダムファイル……………………………………176

リ

リードID………………………………………………181
リプレイス……………………………………………347
リンクポインタ…………………………………22・36

ロ

600ボー…………………………………………………293

ワ

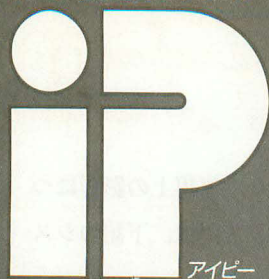
ワークエリア(ファイルコントロール)……………275
1ファイル転送…………………………………………197

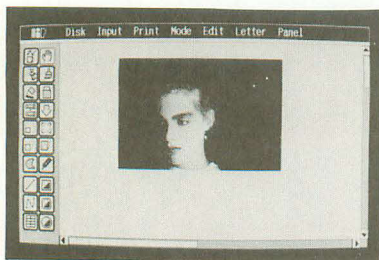
本書に記載されている内容については，筆者らが調査・解析したものであり，運用上の影響については責任を負いかねますのでご了承ください。なお，本書の内容に関するご質問は，下記のシステムソフトまで文書にてお願い致します。

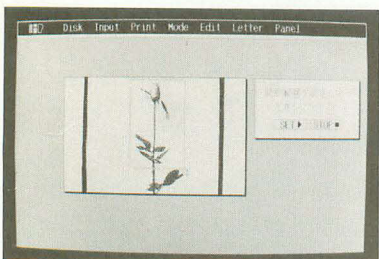
PCファミリー・テクニカル・ノウハウ集
PC−9800シリーズ編
PC−Techknow9800

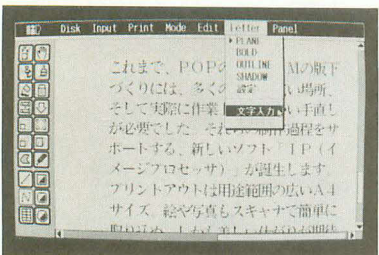
1983年12月　第1版第1刷発行
1988年3月　第1版第11刷発行
定価3,200円

共　著／藤田英時・幸田敏記
監　修／システムソフト
発行者／樺島正博
発行所／福岡市中央区天神5丁目7-2
株式会社システムソフト
電　話　092−714−6236

印刷・製本　日本アート印刷株式会社

印象をこえて、

IP(アイピー)は、これまで面倒だった紙面の編集作業をよりスムーズにする、イメージ・プロセッシングツールです。プリントアウトのサイズはポピュラーなA4。豊富なスキャナに対応し、高品質なグラフィックが作成できます。気軽に使えて、とても便利。多方面にわたってのビジュアル・プレゼンテーションを可能にした、あざやかな表現力です。

お気に入りのフォトグラフはと………

カラー対応のイメージスキャナから、写真をフルサイズで読み込みます。IPでは、スキャナの能力に応じてA4判の大きさまで読み込み可能です。

アクセントが欲しいな!

花の写真は、必要なところだけをトリミングして読み込みます。トリミングは、任意の長方形を指定することにより簡単におこなえます。

ボディコピーはこんなところかな….

書体、サイズの他、文字色や、文字間隔・行間隔を指定し、VJE-βによりワープロ感覚で文字を入力します。

ビジュアルと文字を組み合わせて…と。

文字と絵柄の大きさのバランスや配置などを検討し、A4サイズ(印刷時)の中でレイアウトします。完成したら、プリントアウト。上の写真は、PC-PR801での出力例です。フルカラープリンタでの出力の鮮やかさには目を見はるものがあり、ビジュアル効果を要求される各種プレゼンテーションに威力を発揮します。

コミュニケーショナル。

■機能一覧

描画画面 通常サイズ 640×400ドット
全体サイズ 1192×1752ドット
(印刷時A4サイズに相当・PC-PR201系の場合、18.9×27.8cm
・NM系、EPSON系の場合、16.8×24.7cm)

カラー 基本8色を含む25色(16×16タイルパターン、オリジナルタイルパターン作成可能)
ピンセットで画面上の16×16ドット・セグメントの色をピックアップ可能

ブラシ ブラシ12種
半径・密度の調整可能なエアブラシ
スクリーントーン12種(オリジナルトーン作成可能)
消しゴム
塗りつぶし

基本図形 ライン/スプライン/ボックス/ボックスフィル/角丸ボックス/角丸ボックスフィル/サークル/サークルフィル/多角形塗り/矢印/
表枠(線幅3種、分割数 縦横1～99)

編集機能 拡大/縮小/回転/変形/斜体/切抜き/色変換/カット/コピー/ペースト

モード 全体表示、通常表示、拡大表示、スケール表示、座標表示、方眼表示

文字 連文節変換/逐次自動変換/辞書先読み一括変換(VJE-β標準装備)
書体4種(PLANE、BOLD、OUTLINE、SHADOW)文字サイズ8種
(PC-PR201系の場合10～46級、NM系・EPSON系の場合9～41級)
文字間隔、行間隔、文字色/下地色

ファイル スクラップ(画面上の任意部分の読み書き)
アートマスター400データへの読み書き
MS-DOS標準テキストファイルの読み込み可能

プリント レーザープリンタ(PC-PR406LP)やフルカラーイメージプリンタ(PC-PR801)サポート
A4サイズ固定 マージン指定、印刷枚数指定、印刷範囲指定

スキャナ NEC製 PC-IN501/502/503/503H
EPSON製 GT-3000/V
線密度、読み取りサイズ、読み取り濃度、MONO/TONE/COLOR
※COLORの指定はGT-3000/Vのみ可能

カメラ EPSON製 GT-20
コントラスト、ブライトネス、書き込みモード、MONO/COLOR、書き込みサイズ、バランス(オート/マニュアル)

■必要なシステム

●本体
PC-9801E/F/M/VF/VM/UV/VX
※PC-9801、PC-9801Uでは動作しません。
※PC-9801Eでは漢字ROMが必要です。

●ディスクユニット
PC-9801E/F/VFでは外付の1MBタイプのディスクユニットが必要です。
640KB(720KB)タイプのディスクはサポートしていません。
2ドライブ必要です。

●メモリ容量
本体メインメモリ384KB以上＋1MB以上のRAMボードが必要です。

●RAMボード
1MB RAMカード(システムソフト製)
PIO-9234シリーズ(I・Oデータ機器製、1MB以上のもの)
KR9807-1MB/2MB(加賀電子製)
のいずれか1つが必要です。

●ディスプレイ
専用高解像度ディスプレイ(640×400ドット)

●マウス
PC-9871 MSマウスセット(PC-9801E/F用)
PC-9872/K/L MSマウス(PC-9801M/VF/VM/UV/VX用)
アスキーマウスセット
NEOSバスマウスMS-50
NEOSシリアルマウスMS-40
のいずれか1つが必要です。

●イメージ入力装置
PC-IN501/502/503/503H
GT-3000/V
GT-20(カラー入力アダプタGR-20が必要)
のいずれか1つが必要です。
※シリアルマウス使用の場合、イメージスキャナは
PC-IN501/502/503/503H＋拡張インタフェースボード(PIO-9153、I・Oデータ機器製)
または、
PC-IN503H＋GP-IBインタフェースボード(PC-9801-29N)
または、
GT-3000＋GP-IBインタフェイスボード(#5160、GT-3000に実装)
＋GP-IBインタフェースボード(PC-9801-29K/N)
GT-3000V＋GP-IBインタフェイスボード(GT30VGPIB、GT-3000Vに実装)
＋GP-IBインタフェースボード(PC-9801-29K/N)
の組合せのみ使用可能です。

●NEC製MS-DOSシステム(Ver 2.11またはVer 3.10)が必要です。

●カラープリンタ
NEC PC-PR201CL/HC/H2/V PC-PR801 NM-9950/9700

EPSON ESC/P24-J83・C
VP-2500 VP-2550

SHARP IO-725(タイプI、NMモード)

●モノクロプリンタ
NEC	PC-PR201/H/F	PC-PR101/L/F	PC-PR406LP
	NM-9300/9400	NM-9300S/9400S	NM-5020
EPSON	ESC/P24-J82、83		
	UP-130K	IP-130K	HG-2500
	VP-130K	VP-135K	VP-1000
	VP-80K	VP-85K	VP-800

※熱転写プリンタはサポートしていません。

■メディア
5"-2HD、3.5"-2HD

定価 45,000円